DOCUMENT RESUME

ED 057 597 24 EM 009 441

AUTHOR	Lichten, William
TITLE	Development of New Instructional Uses of the Computer in an Inner-City Comprehensive High School. Final Report.
INSTITUTION	Yale Univ., New Haven, Conn.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C. Bureau of Research.
BUREAU NO	BR-9-A-006
PUB DATE	4 Oct 71
GRANT	OEG-1-0-90006-0101(010)
NOTE	230p.
EDRS PRICE	MF-$0.65 HC-$9.87
DESCRIPTORS	*Computer Assisted Instruction; *Computer Science Education; Digital Computers; Disadvantaged Youth; Inservice Teacher Education; *Programing; *Secondary School Students; *Urban Education
IDENTIFIERS	BASIC (Programing Language); FORTRAN (Programing Language)

ABSTRACT

A three-part program investigated the use of computers at an inner-city high school. An attempt was made to introduce a digital computer for instructional purposes at the high school. A single portable teletype terminal and a simple programing language, BASIC, were used. It was found that a wide variety of students could benefit from this approach. A second part of the program was an attempt to teach the FORTRAN programing language to a select group of gifted New Haven public high school pupils. The Yale IBM 7090/7094 Direct Coupled System was used. The majority of the students mastered all the basic features of FORTRAN during a one-week period. A third part of the program was an investigation of methods of training teachers to use computers as a tool of instruction. Several approaches were used, including voluntary after-school programs, in-service training classes, and use of methods classes and practice teaching sessions in the Yale Master of Art in Teaching (M.A.T.) program. This attempt was most successful with mathematics and physical science teachers. An article on "Introducing the Digital Computer Into the High School Curriculum," a description of a joint teacher-student project to translate slang into standard English, a handbook on BASIC programing, and a short guide to FORTRAN are appended. (JY)

PA 24

EM

OE-BR

BR 9A 006

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ED057597

FINAL REPORT
Project No. 9-A-006
Grant No. OEG-1-0-90006-0101(010)

DEVELOPMENT OF NEW INSTRUCTIONAL USES OF THE COMPUTER IN AN INNER-CITY COMPREHENSIVE HIGH SCHOOL

William Lichten

Yale University

New Haven, Conn. 06520

October 4, 1971

The research reported herein was performed pursuant to a grant with the Office of Education, U.S. Department of Health, Education, and Welfare. Contractors undertaking such projects under Government sponsorship are encouraged to express freely their professional judgement in the conduct of the project. Points of view or opinions stated do not, therefore, necessarily represent official Office of Education position or policy.

U.S. DEPARTMENT OF
HEALTH, EDUCATION, AND WELFARE

Office of Education

Bureau of Research

Region I, Boston, Mass.

EM 009 441

This document was processed for the ERIC Document Reproduction Service by the ERIC Clearinghouse at Stanford. We are aware that some pages probably will not be readable in microfiche or in a hardcopy enlargement. However, this is the best available copy, and we feel that the document should not be withheld from interested readers on the basis of these unreadable pages alone.

CONTENTS

page

Summary 1

Introduction 1

Methods 4

Results 8

Conclusions 17

Appendices. . . . . following 18

I. Copy of the article "Introducing the Digital Computer Into the High School Curriculum", by William Lichten
High School Principal's Service-Operational Briefing, reprinted with permission, copyright Craft Educational Services, Inc. 1970.

II. Text of "The Computer is Stupid", by Franklin Caccuitto and William Lichten, to be submitted to Media and Methods

III. Handbook for a Course in BASIC Programming
Introductory Material
Assignments
Program Library

IV. Yale University - Course in Computer Programming in FORTRAN IV on the I.B.M. 7090/7094 DCS (Direct Coupled System)
A Short Guide

Summary

An investigation was made of the instructional use of the digital computer in an inner-city, comprehensive high school. A single portable teletype terminal was used. The simplest possible language, BASIC, was used. It was possible to reach a wide variety of pupils.

A second part of the program was to teach the FORTRAN computing language to a select group of gifted New Haven public high school pupils. The Yale IBM 7090/7094 Direct Coupled System was used. The majority of this group mastered all the basic features of FORTRAN during a one week period.

A third part of the program was an investigation of methods of training teachers to use computers as a tool of instruction. Several approaches were used, including voluntary after school programs, in-service training classes and use of methods classes and practice teaching sessions in the Yale M.A.T. program. These included using computers for pupils from grades 7 through 12.

It is concluded that the computer can, and should, be used to reach students of all abilities and levels in secondary schools and to make it a significant part of their educational experience. It is equally possible and desirable that a majority, if not all, future mathematics teachers be capable of using the computer as an instructional tool.

It is recommended that the major priority in allocation of resources in the near future should be in the area of educating average pupils in computer skills.

Introduction

This report covers the period of 11-15-1968 to 9-14-1969.

During the 1968-69 school year the principal investigator and author of this report, a professor of physics at Yale University, had a semester leave of absence. During the school year and the following summer the author investigated the possible instructional uses of a modern digital computer primarily in Richard C. Lee High School, although other schools were used.

The Richard C. Lee High School was well fitted in its role to expose all the facets of urban education. The school has a diversified student population, which represents a cross section of present day urban school populations. The school is approximately 50% white and 50% black. The economic and academic background of the students covers the range from desperate poverty to the affluence of the wealthiest section of the city.

The author had experience in computer programming as an aid to his research and teaching at Yale. He was familiar with programming in MAD (Michigan Algorithm Decoder) and FORTRAN (Formula Translator).

Supplemental funds were used to extend the program to train teachers at Lee High School and at the other two high schools in New Haven, Wilbur Cross and James Hillhouse. For short periods computer terminals were installed in these schools also. Terminals also were installed in the Yale-North Haven Summer High School, where practice teachers were given a chance to use the computer in their classes. The latter program was primarily financed by the Yale Office of Teacher Training.

Purpose of the Investigation

As the title of this proposal indicates, the purpose of this investigation was the development of new instructional uses of the computer in a high school. Computers are used commonly in schools for three different purposes:

1- As an administrative tool - Computers are used for scheduling, making up report cards, keeping records and the like.[1] It is the author's opinion that this type of data-processing activity, although it may be economical, contributes little in a direct way to the educational climate of a school. Therefore, it was not a part of this investigation.

2- As a teaching machine - For the reasons described in the appendix,[1] it was decided not to use the computer for computer assisted instruction. A much more massive program would be involved than was possible in the time that was available.

3- As an instructional resource - The possibility of using the computer as an additional resource in the school seemed to be the most fruitful course of action. Since computers are a part of our way of life, it seemed most desirable to give the pupils a direct experience with the computer as a means of understanding their world.[2] The purpose of the investigation then was to investigate the use of the computer for this purpose.

---

[1] See Appendix I, p. 1.

[2] See Appendix I, pp. 1-2.

What is a computer? Computer vs. Calculator - Before a discussion of the methods used in this investigation, it would be helpful to say just what is meant by the word computer, as opposed to the desk calculator which is found so often in schools. A desk calculator can do simple arithmetic: addition, subtraction, multiplication, sometimes division. A computer can do all these with ease, but can do them with much more versatility. The best way to illustrate this is to examine some of the programs which were written during this project.

For example, the program "SPEEDY" calculates the ratio of two numbers. This could be done quite easily with many desk calculators.
However, the computer does more. It rounds off the result to three decimal places (the normal usage in the class that was visited) and gives the result in per cent (also a topic under study by the class). Furthermore, it prints out instructions to the user, which tell him or her how to use the computer. This illustrates the use of a computer as a kind of super calculator.

FACTOR illustrates the greater computational power of the computer. No calculating machine would tell the user that 25,391 is a prime number, and do that in a fraction of a second. It would take hours to solve this program with a calculator.

The use of a computer to balance a check book to print a bank statement to look up material in a table and use it to do complex calculations in science or in business illustrate the versatility of the computer.

At first glance, the introduction of this device into the school introduces a new dimension to education that cannot be met by traditional exposure or by standard methods of teaching.

One valid argument used against the use of computers in teaching traditional subjects is that its breath-taking power at solving problems does nothing to enhance the pupil's own ability to do so. For this reason, the philosophy behind the use of the computer in the high school was that, in its own right, exposure to computers is a valid part of each student's education.

One particularly innovative approach to the use of the computer was used by Mr. Franklin Cacciutto in a slow division of freshman English. He used a program "SLANG", written by the author, which literally translates Slang into Standard English and vice versa. Because of the literalness of its translation, the computer soon appeared "stupid" to the children. Their efforts to teach the computer how to translate their slum patois into respectable English transcended the usual efforts of students at that level. This usage of the computer by a master teacher defies classification into categories.

Another application of the computer was in the field of teacher training, where trainees learned to use the computer in their classrooms. Thus, both teacher and pupil were learning the potentialities of this novel device together. This experiment was carried out at Yale-North Haven Summer High School, which is operated by the Yale Office of Teacher Training.

Methods

Why Time Sharing BASIC was selected - Before starting, it was necessary to make a careful choice among the various types of computers available.[4] Because of the small initial outlay (the program was started with supplemental funds), time sharing was the only feasible choice. It soon became apparent that two features of time sharing were extremely valuable. The ease of the interpreter language, BASIC,[4] was indispensable for dealing with high school pupils and teachers. Furthermore, the power of the giant computers used in time sharing made possible a much more versatile program than would have been possible with smaller systems.[5]

It should be pointed out that large computers actually are easier to use than small ones, yet have a greater range of functions. For example, the ability of BASIC to manipulate words as in SLANG, (See Appendix II), is unavailable on installations that cost less than the half million dollar computer used for this project.[6] Finally, only a minimum of hardware, a rented teletype terminal, was required.

---

[4] See Appendix I, pp. 2-3.

[5] See table at botton of p. 2, Appendix I.

What and how children were reached - It was decided to attempt to reach every kind of pupil in the school, rather than concentrate on children at any particular achievement or grade level. By placing the teletype on a wheeled cart, and by installing numerous telephone jacks throughout the school, it was possible to reach a wide variety of classes. An additional off-line (not connected to a computer) terminal was used for preparing tapes to be used with the terminal.

The following approaches were explored:

1- to give brief demonstrations to a large number of classes, to familiarize a broad cross-section of the student body with the workings of a modern computer. Since Connecticut State law requires two years of science and mathematics for graduation, all freshmen and sophomores took courses in these subjects. Also, the subject matter in these courses lent itself readily to computer demonstrations, such as BACTERIA (biology), CHECK and TAKHØM (consumer mathematics), COMMØN, FACTØR, and L.C.M. (basic algebra), DENSE and TEMP (physical science), QUADEQ (algebra). In addition, a few advanced mathematics and science courses could be reached with MIRRØR (physics), MIXER (chemistry), RTTRI (basic trigonometry) and TRIANG (advanced trigonometry). Approximately fifty class periods were used for this purpose. About half of the school population of 1600 students were exposed to one or more periods of instruction by this method.

2- to use the computer as an instructional aid in the hands of the teacher in the classroom. Brief demonstrations were given by several Lee High School teachers on their own, using the prepared material from the program library. Also, the practice teachers of courses in mathematics at the Yale-North Haven High School found the game and function programs (BATNUM, FUN, MORFUN, XY-FUN and NIM) relevant and useful for demonstration purposes in their classes. In most cases, the teachers limited themselves to using prepared programs. As discussed previously (see p. 4), one teacher used the computer extensively in his freshman English class.

---

[6] Information furnished by G.E. Time Sharing Service.

3- to use the computer as a computational instrument in the hands of largely academically, unmotivated pupils in regular classes. One mathematics teacher, Mr. David Howell, used the computer terminal on a daily basis in a consumer mathematics (non-college bound) class. In it, the students used it as a desk calculator, but were encouraged to learn to write programs of several lines length to solve classroom problems or to do homework.

4- to allow average students to learn how to program the computer. Approximately one dozen average (largely non-college bound) students came to informal computer classes. Most were recruited from a group of students taking electronics. A few came because their interest had been aroused in the consumer mathematic class.

There were problems in getting these students to come. Very few of them would come after school because of jobs or other reasons. They came once a week during their study hall periods to the computer terminal, where they were given instruction in small groups by the author. During the second term, when this type of instruction was no longer feasible, a school activity period made it possible for students from this group to continue in the Computer Club.

5- to use the computer as a medium for gifted students. A group of about one dozen Lee High School students (almost entirely college-bound) attended an informal after school computer club. The self-selected group was largely composed of the brightest, college-bound students. The first four meetings consisted of a one hour lecture-demonstration, followed by a laboratory period lasting for about one hour.

After passing a simple test of ability to handle the terminal, all students were allowed to sign up for it at any period of the day, or after school.

6- to teach a group of gifted pupils FORTRAN, an advanced programming language. A group of ten gifted students, drawn from all three New Haven high schools were selected on the basis of their ability at computer programming, as demonstrated by their previous mastery of

BASIC. They took a five day course during school vacation in programming in FORTRAN-IV, a language commonly used in science and engineering.[8]

Instruction was largely of the programmed nature, with a series of tasks to be completed as evidence for learning the material. The amount of time put in by each student that week ranged from 15-80 hours, depending on the interest and enthusiasm of the individual

Teacher Training - Three classes were set up for teachers to learn programming in BASIC.

1- The first was an informal, after school class with a one hour lecture-demonstration, one hour laboratory format. There were approximately ten teachers who took this course.

2- The second course was a more formalized in-service training session, with eight participants who were secondary school teachers in the New Haven system. They were selected jointly by the author and by supervisory personnel of the New Haven Schools. A stipend, paid out of supplementary funds, was given to each participant.

The group met for four hours each Saturday during the month of March. The curriculum was of the programmed variety, with assignments to be completed on the computer.[9]

In order to accomodate all members of this course without delay, there were three on-line (connected to the computer) terminals and two off-line teletypes for preparing tapes.

3- A third course was a similar 20 hour computer laboratory for a course in methods of teaching mathematics for six practice teachers in the Yale M.A.T. progeam. During the same period that they were taking the course, the teachers were giving their first courses. An additional ten hour, abbreviated course was offered for four science teachers in the M.A.T. program.[9] Two on-line and an off-line terminal were provided.

---

[8] See Appendix IV.

[9] See Appendix III.

Results

The raw material of education is people, and it is upon the effect on individuals that we must evaluate educational programs. For this reason, the results of this program on a few sample cases will be presented before a more general evaluation is made.

Leonard H. was a senior who was a National Merit finalist. His ability was such that he could do anything. He joined the computer club as one of his numerous after-school activities. He spent very little time on it until the author suggested that he use BASIC to set up the headlines for the SCOPE, the high school newspaper. Justification of type is a lengthy process to do by hand. Leonard wrote a successful program to do this which involved the most advanced parts of time-sharing BASIC. He also took the FORTRAN course that spring.

The following summer, he was hired (with supplemental funds) to assist in the Yale-North Haven summer program. His duties were to assist teachers in giving demonstrations to their classes, act as a laboratory assistant during the teacher training classes, and to fill in his time by writing programs. He is the author of the Function Programs (M0RFUN, XY-FUN). Not only did this bright student benefit from the program, but he contributed a great deal to it.

Clarence B. was a member of a special education class (for pupils with I.Q.'s less than 70) that was visited several times by the author. He happened to attend during a session that a terminal was wheeled into his class. Despite the fact that he missed a briefing session, he unhesitatingly sat down at the teletype, used the one page instruction manual[10] and solved several arithmetic problems. (The runs for these problems are shown on the next pages.)

---

[10] HOW TO WRITE A COMPUTER PROGRAM IN BASIC, see Appendix III.

Clarence B

19B

```
SYSTEM--BASIT←CC
NEW 0R 0LD--NEW
NEW FILE NAME--HER0LD
READY.

10PRINT 46 +89+13
20 SE←←D←END
RUN



HER0LD     10:37    M   TUE 01/07/69

 148



USED        1.50 UNITS.
10 PIN←←RINT 23-14
RUN



HER0LD     10:40    M   TUE 01/07/69

 9



USED        1.50 UNITS.
10 PRINT 3*23
RUN



HER0LD     10:43    M   TUE 01/07/69

 69



USED        1.50 UNITS.
10 PRINT 48/
10 PRINT 48/3
RUN



HER0LD     10:45    M   TUE 01/07/69

 16
```

Clarence B.

(19C)

```
10 PRINT 3/4
RUN
WHAT?
1←RUN


HERØLD      10:48    M   TUE 01/07/69

 .75



USED        1.67 UNITS.
10 PRINT 27*381
RUN


HERØLD      10:51    M   TUE 01/07/69

 10287



USED        1.50 UNITS.
10PRINT 16?
10 PRINT 16/256
RUN


HERØLD      10:54    M   TUE 01/07/69

 .0625



USED        1.50 UNITS.
10 PRINT 256/16
RUN


HERØLD      10:56    M   TUE 01/07/69

 16



USED        1.67 UNITS.
BYE
```

The next day Clarence failed to show up for his second and final chance to use the computer, despite the fact that he was seen in the halls of the school that day. More than ten years of special education had created a gap, that a few classroom visits with the computer failed to bridge.

Timothy C. was a mathematics teacher in another New Haven high school, without any previous experience. He attended the informal class for teachers at Lee High School. He picked up a computer manual, studied it one evening and wrote his first program, to factor an integer into its primes. It worked immediately, after a trivial "bug" was removed. His success at learning programming without formal instruction was quite impressive. Because of his promise, he attended the in-service training sessions. He refused to keep the money he recived for these sessions, but contributed it towards the cost of computer time-sharing at his school. The program at Hillhouse was the most active teacher-run program in all three high schools.

Brian W. was a Yale M.A.T. student teacher in chemistry and physics. He had over a year's experience in a job as a computer programmer. Despite this background, he made less progress in the computer laboratory than the other science teachers, who had received no training in computer programming during their training as biology majors. He was equally unsuccessful in the other aspects of his teacher training and flunked out of the program.

Roscoe T. was a burly, tough, athletic sophomore at Lee High School. He wanted to transfer to another high school because, as he explained it, he liked fighting and "They don't let you at Lee High." He was repeating freshman mathematics, having failed it the first time. He turned up for the informal student classes and attended regularly, even though most of the discussion seemed over his head. He had trouble pressing the keys of the teletype, because of the immense size of his ham-like hands. Nevertheless, he persisted and learned enough programming to do his homework on the computer, much to the surprise (and disbelief) of his teachers. Although his programs lacked the sophistication of the more gifted members, they go beyond using the computer as a mere desk calculator. (See the attached list.)

ROSCOE

```
TAPE
READY.


10 PRINT "HERE IS YOUR HOMEWORK FOR YOU"
20 PRINT "A"            "L"
30 PRINT
40 READ L
50 LET A=89421*L
60 PRINT A,L
70 GO TO 40
80 DATA 9,4,6,7,8,2,1,
999 END
KEY
READY.

RUN
WAIT.



TAYLOR    14:42   M  WED 01/15/69


INCORRECT  FORMAT     IN   20





USED       1.00 UNITS.
20 PRINT A DELETED
20 PRINT"A","L"
RUN




TAYLOR    14:43   M  WED 01/15/69

HERE IS YOUR HOMEWORK FOR YOU
A              L

 804789         9
 357684         4
 536526         6
 625947         7
 715368         8
 178842         2
 89421          1

OUT OF DATA  IN 40




USED       2.00 UNITS.
GOO

*** OFF AT 14:44   ELAPSED TERMINAL TIME =      3 MIN.
```

General Evaluation

1- Demonstrations to large number of classes. This aspect of the program was one of the most successful. Despite the fact that good conduct does not come automatically in an inner-city school, in the overwhelming majority of the classes, interest was high, intelligent questions were asked and the atmosphere was one of appreciation of the importance of understanding how the computer works.[11] About 90% of the pupils paid attention to the speaker, presumably because of the high level of interest and novelty of the subject matter.

The amount of understanding of what a computer is and can do covers an enormous range in a comprehensive high school like Lee. In each class there was usually one or two people who understood enough about the computer to answer questions. It was not unusual to find a person who really believed that the teletype contained a little man inside. In the opinion of the author, the effect of just a short exposure to the computer, if well presented and if it reaches a broad audience, is just as valuable as the intensive experience obtained by the brighter, more capable students (see #3 below). Several serious students told the author that they intended to go into computer technology as a vocation after seeing demonstrations that aroused their interest.

There are certain limitations that go with these demonstrations. It was found that only a small group of people can watch a teletype terminal at one time. For larger classes of 25-30 students, as occur at the Yale-North Haven High, it was found desirable to use two terminals for demonstrations, with the necessity of a student assistant to help the teacher.

2- Use of the computer as an instructional aid in the classroom. It took an exceptional teacher to have the time and creative power to develop his or her own programs for use of the computer as an effective aid to classroom teaching. However teachers readily accepted use of the computer, if prepared programs were available. Interactive, or game-playing programs, such as FUN, M0RFUN, XY-FUN and SPEEDY were particularly successful. If some assistance, such as a department head or student were available, teachers readily used the computer on their own initiative in their own classes, both at Lee and Yale-North Haven High School.

---

[11] In only one class out of the total number of about 50 was there a conduct problem. There the students almost came to blows over who was to have the first turn at the terminal.

3- Use of the computer as a computational instrument in the hands of academically unmotivated pupils in regular classes. This attempt was of very limited success, for several reasons, some of which were personal. The single teletype terminal was inadequate to meet the needs of even that portion of the class that chose to use the computer within the limited time that was available. However, a few members of the class became interested and appeared at the computer club meetings during activity period. That the computer succeeded in getting the interest of these individuals is a positive accomplishment.

4- Allowing average students to learn how to program the computer. With the exception of the small number of pupils, like Roscoe T., who took the initiative in learning to program, the bulk of this group of students was content to learn what they could during the scheduled hours of their informal computer classes. As a result, progress was slow, and only a little was learned during the 5-10 periods that were spent. Here again, limited time and facilities precluded substantial progress.

5
and
6- Use of the computer for gifted college-bound students. This group of students had, or made, time to learn the material. They were well motivated by virtue of their self-selection. They were accustomed to, and capable of, independent study of the materials necessary for learning programming, such as the manuals provided by the computer company. They had sufficient mathematical background to learn practically all of the repetoire of the BASIC language set. As has been found in other high schools, the top students are always quick to take advantage of the computer, and are capable of learning to program much more rapidly than their teachers.

During the spring recess, the group of ten students who were selected from a city wide pool of students took the FORTRAN-IV course at Yale, given by the author. Because of their obvious ability to study independently and the absence of competing school activities, the programmed format of the course was extremely successful. By the end of the week, the majority of the group, as judged from actual performance on the computer, had learned all the essentials of this difficult language, including FORMAT, GO TO, IF, DIMENSION, DO, FUNCTION and SUBROUTINE statements. Only a few of the finer details of FORTRAN, such as EQUIVALENCE, COMMON and DATA statements were not reached by this group. The rapid progress of this group was impressive.

Other Results

Formal vs. informal instruction

The success of the various components of this program seemed to be directly related to the informality and lack of structure in the instruction. That this is true depends partly on certain universal aspects of education, such as self-motivation being the best spur for learning. However, there were certain special aspects of the computer that made independent study especially productive. These were largely the limited capacity of the terminal which made for individual, rather than group instruction. Also, the rate of learning programming is a skill which seems to vary too widely among individuals for group instruction. Finally, the computer acts as a teacher, administering gratifying results if the program is correct, and frustrating "error messages" if the program is wrong. For these reasons, programmed instruction seemed most successful, at least with the more talented and eager students.

Bright vs. average students

The bright students almost immediately mastered BASIC and, with the exception of those who were going on in science or engineering, equally quickly dropped the subject and went on to other interests. The less bright students took much longer to generate interest and learned much more slowly. The effect of this on the composition of the coterie of computer fans was quite marked. At first, the computer club consisted almost entirely of bright, future college students, with a few average or slow members. As the school year went on, the Lee High School Club changed to largely an average or slow group, with a few bright students acting as hangers-on and general helpers. The process also repeated itself at Hillhouse to a more limited extent, as the period in which the terminal was in the school was shorter.

Subject matter and enthusiasm

With the exception of a few applications to language and music, the world of computers centers on mathematics, science and business. This was partially reflected in the reception of the computer by faculty and practice teachers. In general the reaction was as follows:

Mathematics, Physics, Chemistry and Physical Science teachers could apply the computer to their subject immediately and received the computer well. A strong majority of the teachers and pupils in these subjects were eager to use the computer in their classrooms, and only an occasional individual among the faculty or their students reacted negatively.

Biology and Social Science teachers showed an interest in the project but, because a connection with their subject was not established, no strong relationship was established.

Business Education personnel up the line from teachers through department heads to supervisor treated the computer with apathy or outright defensiveness. This was despite the fact that the author developed several programs for letter writing, sorting, filing (see A2Z and NUM), and other business applications (see CHECK, TAKHOM, and DATA PROCESSING). This lack of interest is related to the limitations of business education in the New Haven (and many other) public schools. Bookkeeping, typing and keypunching are the extent of courses given. The courses in consumer mathematics are a good example of interesting material which was taken over by the mathematics department by forfeit. As a result,business ed. has become less relevant to the modern job market and unpopular with high school students.

Effects on the Schools and Individuals

As a result of this program these are the following long-lasting effects on schools and individuals:

1- Lee High School has purchased a DEC PDP-8 computer and has installed it in a mathematics laboratory for use by pupils and teachers.

2- Hillhouse High School will keep a time-sharing terminal in its building as a permanent part of the program.

3- As a result of classroom demonstrations, hundreds of high school students have been exposed to the computer and have been made aware of its potentialities and limitations.

4- At least three students in the program have learned sufficient programming to obtain jobs based on these skills. These were all college-bound, however.

5- Approximately a dozen teachers have received intensive computer programming training. They and a much larger group have had experience incorporating the computer as a part of classroom instruction.

CONCLUSIONS and RECOMMENDATIONS

The computer is bound to play an important role in high school teaching in the near future. As a result of this project, coupled with observation of similar projects in other schools, the following recommendations are made:

1- Education of gifted students. Gifted pupils are quite capable of learning computer programming in high school and should not have to wait before reaching college. They can be reached with a minimum of equipment and formal instruction. The systems commonly used in high schools[13] are quite adequate for this group. The practice of giving these pupils formal courses is of questionable necessity. They can teach themselves.

2- Education of average pupils. The average student who goes on to a job, to a junior college or other form of post-graduate education can profit from the vocational opportunities in the data processing industry. This group is not being reached at present in most high schools' computer education. Efforts to reach this group will involve more extensive efforts than are in force today:

  A- Computers. More elaborate systems than in common use today[13] are required. The critical features are languages available,[13] number of pupils that can be simultaneously accommodated and cost per pupil. Small computer and programmable electronic calculator manufacturers are developing systems that may meet this need at reasonable cost.

  B- Personnel. Although in-service training is a valuable component of teacher preparation, a great deal can be done with present staff, if assistance is given to them. This involves maintenance of a program library in usable form, on tapes or stored in the central computer, which will be relevant to the material given in the classes. Probably this could be handled by an effort comparable to that given to audio-visual services in schools--part of the time of one teacher with several student assistants would suffice for each school. Every pupil should have the benefits of a computer demonstration in at least one class per year.

---

[13] See Appendix I, p. 2.

Development of high school curriculum for courses in data processing for this group of students is needed more than any other activity in computer education. It is a major effort which will require heavy support by educational agencies. It should be the highest priority of any funding in this field.

C- Maintenance. Because of the complexity of computer equipment, school personnel should not be held responsible for maintenance. Service contracts are the normal way of handling this problem.

3- Education of slow pupils. Use of the computer, or electronic calculator, should be explored for this group other than as a teaching machine. There is some evidence that at least some members of this group can be treated as more than objects to be manipulated by the educational system. Mastery of a skill, such as using a calculator keyboard, might have a more positive effect than the effect of unremitting drill of elementary school subjects.

Teacher Training

The average mathematics teacher has the ability to understand and master the art of computer programming. There is no reason why all mathematics teachers should not have acquired this skill as part of their college, or post-graduate education. Hopefully, future teachers of other subjects, such as science and business, should also acquire this skill. For the present, it appears most realistic that the backbone of high school computer education will be in the mathematics departments.

HIGH SCHOOL
PRINCIPAL'S
SERVICE

OPERATIONAL BRIEFING
APRIL • 1970

"PERMISSION TO REPRODUCE THIS COPYRIGHTED MATERIAL HAS BEEN GRANTED BY Croft Educational Services, Inc.

TO ERIC AND ORGANIZATIONS OPERATING UNDER AGREEMENTS WITH THE U.S. OFFICE OF EDUCATION. FURTHER REPRODUCTION OUTSIDE THE ERIC SYSTEM REQUIRES PERMISSION OF THE COPYRIGHT OWNER."

Introducing the Digital Computer Into the High School Curriculum

By William Lichten

Every high school should have its own computer—and every interested student should learn how to use it.

But computers are frightfully expensive, you're probably thinking. Some are, it's true, but some are not.

Well, then, aren't computers much too technical and complicated to be understood and used by high school students? Not at all.

So does a high school student need *to know about computers?* Definitely. If he goes on to college, he'll probably have to use one. If he goes into business, banking, transportation, manufacturing, research, science, engineering, or government, he'll need to know about them. To be ignorant of computers, in fact, is to be unprepared to live in today's automated society.

To repeat, then, every high school should have its own computer. And the prime questions to be considered are, what computer system to acquire, and what to use it for. We'll explore these questions as we proceed.

Not an Administrative Tool

First, abandon any notions you may have of using your computer to perform administrative tasks.

Most administrative work such as grade reporting and attendance keeping does not require the super speed and brains of an electronic digital computer. Such work can usually be done on simpler equipment, like the accounting machines you already have, and any attempt to delegate it to your computer will probably impose severe limitations on the instructional uses to which the computer can be put.

Scheduling, on the other hand, is definitely a job for a computer. But the best job of scheduling is done by a very large, fast computer using an expensive program written by experts, and it probably will not be feasible for you to schedule classes on any computer you can afford to buy or rent. If you wish to use a computer for scheduling—and computer scheduling can add a new dimension to your educational program—go outside to a service bureau.

Not a Teacher

Abandon also any notions you may have of using a computer for so-called computer assisted instruction (CAI). There are two major forms of CAI—one in which the computer serves as a tutor on an elementary or remedial level, usually, and another in which the computer serves as a data bank or information retrieval system. Both forms of CAI require programs costing hundreds of thousands of dollars to prepare, and it is unlikely that you will be able to develop them on your own.

An Instructional Resource

For what purpose, then, could a computer be used in your school? For the purpose of teaching students how to use it.

A computer is a super calculator of tremendous versatility. Students of science and mathematics can use it for working out lengthy and complicated calculations. Students of social studies can use it for testing theoretical models—for example, predicting voting patterns in an election. Courses in computer programming, computer science, and computer applications can be developed and offered if students have access to one of these versatile machines.

© CROFT EDUCATIONAL SERVICES, INC., 1970

If you have the right computer, *any* student can learn to use it in a few days. In three or four lessons the bright college-bound student with a scientific and mathematical bent can learn to use a computer for working out complex calculations. In the same amount of time, the very slow student can learn to use a computer for working out low-level arithmetical problems.

But you must have the right computer.

As is the fate of many new programs, a poorly planned computer installation can be disastrous. Costs can escalate out of sight. Teachers can resent or fear a computer and refuse to work with it. Or the language a computer uses can be so difficult as to be understood only by a few bright students and perhaps one teacher, who can leave you with a room full of idle equipment if he decides to seek his fortune in another school district.

Thus your choice of a computer system must be made with great care–preferably in consultation with objective experts. But most of what you need to know to make a sound choice is contained in the remaining columns of this paper and in the chart below.

Basic Understandings

A computer operates on the basis of a program, which is nothing more than a set of instructions. The program may be highly complex, consisting of thousands of separate instructions . . . or it may be very simple, consisting of two or three instructions. Depending on how the computer is constructed, the program may be fed into it by means of punched cards, a tape, or a keyboard terminal that looks like an electric typewriter. Two or more of these methods can be used with some computers.

A program must be written in a language that the computer understands. If the computer is to be practical for high school use, it must employ a language that the student can easily master.

Four types of languages are used with computers.

Machine (or binary) language is the language of the computer itself. A digital computer is really a gigantic arrangement of electronic switches, which can be either on or off, and machine language is being used when the computer internally, or the operator externally, is manipulating the switches. For the operator to use machine language is a very slow process, but when the operator uses any other language, the computer must translate his instructions into machine language before it can proceed.

All practical computer programs are written in words in a symbolic language, and of these languages there are three types:

Assembly language (a symbolic language) is the closest thing to machine language, and therefore makes the most efficient use of the computer. Experienced programmers use assembly language, but its use is a slow "off-line" process that would strain

At A Glance - Five Computer Systems For Possible School Use

Typical High School Usage	Language Used	Sample System	Approximate Cost	Number of Pupils With "Hands On"†	Cost Per Pupil With "Hands On"†
		Computers Outside School			
batch-processing, programming	assembly, compiler (RPG, FORTRAN)	IBM 1401, 1130, etc.	$25-$50/hr.	25	$50/yr.
time-sharing (terminal in school)–for all levels of math and science	compiler and interpreter (BASIC, FORTRAN, etc.)	GE Time-sharing service, 1 on-line, 1 off-line teletype	$10/hr. + teletype rental	25-50	$100-$200/yr.
		Computers Inside School			
all levels of math, science; computer science & programming	machine, assembly, compiler (FORTRAN), interpreter (FOCAL)	Digital Equipment Corp. PDP-8 + fast reader + off-line teletype	$13,100	25-50	$250-$500
	machine, assembly, compiler (FORTRAN), interpreter (BASIC)	Hewlett-Packard 2114A (8K memory + high speed reader + 2 teletypes)	$20,550	25-50	$400-$800
advanced math, science, computer science & programming	machine, assembly, compiler (FORTRAN)	IBM 1620 + key punch (school must have card reader, air conditioning)	$8500/yr.	10-20	$400-$800/yr.

†Based on a full school year program. More pupils can be reached by demonstrations, or if classes take turns.

the patience of a beginning computer student. Bright high school students can learn it, however.

Compiler languages (symbolic, algebraic) are the most widely used computer languages, but they are difficult, requiring slow off-line compilation—so slow even as to strain the patience of the most experienced programmer.

Interpreter languages (symbolic, conversational), on the other hand, make using a computer so simple that even the most backward of your students will pick it up. The reason for the success of the interpreter are as follows:

1. "Desk calculator" ease in writing statements. It is just as easy, if not easier, to write in simple interpretative language as it is to solve a problem on a desk calculator.
2. Conversational mode. Using a computer with an interpreter is like having a conversation. The computer asks what you want, you type in your orders, and it gives you back your answer immediately. If your orders are in incorrect language, the computer tells you what's wrong.
3. Freedom of annoying details of format. With most compiler languages, input or output instructions must be at precisely the correct location on the punched card or line of the typewriter. Format has no place for teachers and students who don't intend to become professional programmers.

You've probably heard of such computer languages as FORTRAN, BASIC, COBOL, FOCAL, and RPG. These are names for some of the languages listed above, but you needn't pay them much attention. *The important thing to remember is that the computer you select must be built around an interpreter language with the three features outlined above. This is indispensable.*

Getting Started

There are three ways of getting started.

1. Batch-processing. In batch-processing, you make use of someone else's computer. You plunk a pile of IBM cards on a counter and are charged only for the time it takes the computer to process them. Not only is it the dirt cheapest way to pay for time, but you have none of the worries of owning and operating a computer. If you are near a city of any size, there may be a commercial computer service bureau, a business with a computer that is idle part of the day, or a university that will be willing to sell computer time to you. You might even be able to get some time on the computer in the superintendent's office.

If your school already has a set of key punch machines, the only expense of batch-processing will be the cost of the computer time, if any. Another advantage of this method of using computers is that your students will be able to write their programs in the languages actually used in business (BAL, RPG, COBOL, etc.) or in science and engineering (FORTRAN).

The chief disadvantage of batch-processing is its catch-as-catch-can nature. The company that is willing to let you use its computer for an hour a day this year may decide to sell it next year. The computer used by the superintendent or the city government may become so overloaded that students will have trouble getting to it. Another disadvantage is the long "turn around time" (waiting) to get computer results.

2. Time-sharing. This method too involves the use of someone else's computer. A terminal (usually a teletype machine the size of an electric typewriter) is installed directly in the school and is hooked to a remote computer by telephone lines. The school pays only for the actual time used on the computer, even if it is only a fraction of a second.

Time-sharing can be started on a small amount of money ($500-$1000 can sustain a small program in a school for several months). Time-sharing is cheaper in the short run (up to three or four years) than buying. The terminal takes up less space than any other form of computer installation, and you can cheaply install several telephone jacks in different parts of the school, put the teletype machine on wheels, and roll the computer terminal into the classroom for a demonstration.

Time-sharing is the fastest of all computer services, being capable of compiling and assembling programs in a few seconds. Almost all services have an easy interpreter language, in addition to the usual compilers, although machine and assembly languages usually are lacking.

The main disadvantage to time-sharing is that careful supervision is required to control expenses. A very complicated set of charges goes with time-sharing, and an unskilled, unsupervised user can run up amazing bills in a short time.

3. Your own in-house computer. It is possible to have your own computer under several arrangements— purchase, rental, or lease. The advantages are no hourly charges and unlimited, immediate access. Some computers are small enough to be moved from classroom to classroom. Unlike time-sharing and batch-processing, the student has a chance to operate the computer himself. He can try out machine and assembly language at his convenience.

Mathematics textbooks written in FOCAL and

BASIC are available, and these make it easier to tie computer use into the mathematics curriculum. Also, small computer manufacturers usually supply good software, including a free course on use of the computer.

The disadvantages of buying a computer are difficulties in getting funds. Another common headache is a large, unexpected repair bill. An in-house computer is also subject to fire, vandalism and, to a lesser extent, theft.

Get Advice

Before venturing far, get the advice of such individuals as the following:

1. Knowledgeable people in your school system.
2. Colleagues (principals, department heads, or teachers) *in your immediate area* who have used computers.
3. Representatives of local educational or commercial data processing installations.

These people can tell you which instructional activities will best prepare your students for post-graduate opportunities. More important, the interest you generate may get you free help in starting your program.

Getting Over the Financial Hurdle

It's hard to get a school board to plunk down large sums of money for a program that is completely untested in their eyes. There are ways of getting around this, however:

1. It may be possible to get started with no cash outlay by taking advantage of a generous computer salesman or business firm, or by latching on to a government-financed university program. These arrangements usually last only a short time, however, and sometimes don't give you the type of computer you want. You must be very careful not to allow these gifts to tie your school down to a system you don't want.

2. You can rent or lease equipment. In the data processing industry it is common to rent by the month equipment that is too expensive to purchase outright. Some companies lease equipment and allow you to apply most of your rental towards purchase of the computer at a later time. If you rent or lease, be sure you have a clear agreement with the lessor as to who is responsible for maintaining, repairing and insuring the equipment.

Four Tips for Success

Successful programs of computer instruction will include the following:

1. Computer clubs—a sure fire way of getting started in your school. Invariably, these attract the students most able to learn to operate the computer successfully.

2. Inservice faculty training. One good teacher can get computer activities started. However, without a wider base among your teaching staff, computers will not receive the use that justifies their expense.

3. Use in the regular curriculum. Computers or calculators can be used throughout mathematics, science, and in computer science and programming courses. After-school activities can be conducted for junior high schools or other high schools that do not have the same facilities. The more students you can reach, the better you can justify your costs.

4. Publicity. Once your program has gotten off the ground, demonstrate it to visitors, PTA meetings, and to the local press. A computer curriculum in your school is excellent for public relations. Let the public know what it's getting for its money.

A CHECKLIST: Questions to Ask When You Buy or Lease a Computer

1. What kinds of students can be reached with this equipment?
2. What subjects can be taught with this equipment?
3. For what type of job will its use prepare my students?
4. What languages does the equipment understand?
5. What kind of training will my teachers need to use this equipment?
6. What do I need to provide in the way of special facilities for installation? (For example, how much space is required? Are air conditioning, special power or telephone lines, or flooring required?)
7. Can the equipment be used in a regular classroom? (Consider the factors needed to answer question No. 6, especially factors such as portability and noise produced by the equipment.)
8. What additional equipment is required to go with the basic units? (For example, some computers require teletype machines; others require key punches.)
9. How much will the total package cost, including maintenance and insurance?
10. How many pupils will be serviced by it and what is the cost per pupil?
11. What are the various methods of paying for the equipment, and which are best suited to my needs?
12. Who will maintain the equipment and repair it if it breaks down?

William Lichten, *professor of physics at Yale University, has taught computer programming to high school and college students and teachers.*

The Operational Briefing is a component of the High School Principal's Service, published by Croft Educational Services, Inc., and is circulated privately to educators by subscription. General Offices: 100 Garfield Avenue, New London, Connecticut. Reproduction in whole or part without permission of the Publisher is forbidden. Printed in U.S.A.

"The Computer is Stupid!"

Using a Computer in an inner-city English Classroom

By Franklin C. Cacciutto and William Lichten*
Richard C. Lee High School,
New Haven Public School

Slang and the Student

All living languages change and develop, but relative to standard English, slang is evanescent. The meanings of slang words change constantly and from place to place. According to the Dictionary of American Slang, in 1885 the word dude meant "1. an overdressed man. Perhaps from or related to duds = clothes. → 2. A man from the East or a city man vacationing on a ranch." Today, in the argot of the Eastern inner-city, a "dude" is one of the gang, a regular guy, admirably normal and adjusted. Application of the term applies acceptance, in contrast to the connotation of rejection the term once carried.

In part because of its energy and pace, slang is one of the most poetic and creative sides of spoken English. Among slang groups, teenager slang changes most rapidly. Among teenager slang groups, the slang of the inner city teenager changes most profoundly and energetically. In the inner city words appear and fade from currency almost weekly. Some words seem elusive, even ephemeral in meaning, as they flow through the duration of their short life. Such words are caught in the tensions of the honesties, ambivalences, and uncertainties of ghetto life.

For example, the word "up tight" can mean both "in trouble,"

---

*Permanent address, Yale University. This research was supported by grants from the U.S. Office of Education and the New Haven Foundation.

as in "He tried to pass and now he is uptight about the course," and also "close to," as in "He is uptight with the crowd." Applied to the relationship between a boy and girl, as in "He is uptight with that chick," the term takes on remarkable tensioned significance. Slang words make interesting artifacts, but in the living slang group there is no delusion that the word is the thing. For example, the word "conservative" is often applied as an honorific, a compliment referring to dress style and demeanor. The apparent opposite of the word "conservative" is the word "jitterbug," which is applied in warm and even friendly depreciation of a person whose dress style and demeanor are extremely frenetic and colorful. However, when pressed for the meaning of "conservative" in this context, Larry Bolden, a Freshman, refused to render the obvious value judgement implied. He retorted that "It's just a word." The seeming opposition between "conservative" and "jitterbug" was resisted.

As Wentworth and Flexner have observed, "among the teenagers, entertainers, college students, beatniks, jazz fans, intellectuals, and other large slang groups, slang is often used in preference to standard words and expressions. Slang is the 'official' modern language of certain vociferous groups in our population."[1] The inner city has its own argot, a qualitative language style which serves to help identify the speaker. Thus, we murder to dissect. When inner-city slang, or, indeed, any slang, is captured in literature or enters that in-between group of words that is neither proper nor dispensible, it tends to move out of the domain of current slang.

Such slang can become the object of lexicographers. Both the

[1] Harold Wentworth and Stuart Berg Flexner
Dictionary of American Slang, (Thomas Y. Crowell, N.Y., 1960)

Random House Dictionary, which is unprescriptive and, therefore, less a conventional dictionary than an inventory of words, and the Dictionary of American Slang, edited by Harold Wentworth and Stuart Berg Flexner, deal with "slang" that is here to stay. But slang in the argot of the more individualistic groups is less easily compiled; it changes constantly by virtue of its hermetic, identification function, and cannot be captured in a list.

Because slang tends to transmute into artifact as attention is paid to it, no final dictionary of slang is compilable. Slang is elusive and ephermal and evanescent, much like youth itself. What is fresh and stylish "jive" one month seems stiff and awkward the next.

The Computer

Since the use of the computer in teaching English in an inner-city high school is somewhat of a novelty, if not unprecedented, it is perhaps worthwhile to devote part of our essay to explanation of what a computer is and what it does. If the reader were seated at the rear of the Lee High School classroom, he would see the door open and watch a teletype terminal be wheeled into the classroom. In appearance, the teletype machine appeared to be only a slightly oversized electric typewriter. Immediate reaction of the student was that "the computer" was being brought into the classroom. We first had to explain to them that the teletype machine was not a computer; the computer was elsewheres, for reasons of expense: and portability. Even the smallest and least expensive digital computer costs in the neighborhood of $10,000. This was far in excess of our entire budget for the year. The cost of a really useful, electronic, digital computer, in dollars, runs to 6 or more figures. Obviously, neither Yale University nor the New Haven School system was about to commit this type of expenditure to our small project. Even if one were to use someone else's computer and just pay for the time, by the hour, the cost typically runs between $100-$1,000 per hour. This also was beyond our reach. Secondly, the giant electronic computer, even though micro-electronics has performed miracles of miniaturization, still occupies the greater part of a rather large room. Faced with a choice of filling the classroom with a huge computer or with students, we chose the latter. How then did we solve the problem of bringing the services of a giant, immensely expensive digital computer into the classroom?

The secret of our project was the use of "time-sharing" computer technology. This ingenious technique, originated at Dartmouth College by Professor J.G. Kemeny, allowed the Lee High School students to sit at the teletype machine and use the computer as if it was their very own, although in fact many others were using it at the same time. As we explained to the students, the computer acted like a busy waitress behind the counter in a restaurant. Just as she rushes up and down the counter taking each dinner order, relaying them to the cook bringing the dishes back to the customers, answering questions and performing all the other innumerable services that waitresses do, the computer likewise takes care of many customers all at once. Occasionally when we typed in a command to the computer it said "wait". The "wait" was hardly more than a few seconds in almost all cases. Thus, we had the illusion that the computer was all ours and devoted just to our needs. In reality, as many as 40 other users from the State of Connecticut had their teletype machines connected to the computer (which was in Boston, Mass.) and were receiving the same illusion that it was all theirs.

To understand how this excellent service is achieved via time-sharing, we'll have to examine a little more closely how a computer works. The user, a Lee High School student in our case, sits at a teletypewriter, types in his commands to the computer, and watches the electric typewriter type out the computer's questions and answers. The teletypewriter is connected via telephone lines to the computer. In the more technical language of the computer engineer, the teletypewriter is acting both as an input and output device. The computer itself consists of three parts: the memory, the control unit, and an arithmetic and logical unit capable of doing arithmetic and, more important in our case, capable of manipulating symbols, such as numbers, letters, words and even sentences. For purposes of brevity, the computer (control, memory, arithmetic and logical unit) often is called the central processing unit, CPU. The secret behind time-sharing lies in the incredible speed of the CPU. Most students, seated at the console, type at a rate of only a few words per minute. Even if we use a device attached to the teletype machine which punches a paper tape and then reads into the computer, the rate of transmission of information into and out of the computer is limited by the teletype machine itself. This amounts to about 10 characters (letters, numerals or punctuation marks) per second, or about 100 words per minute. The computer, being much faster than the teletype machine, can receive many times as many words per minute as our stammering teletype machine manages to transmit. Not only this, but the CPU can take these instructions, store them in its memory, do whatever tasks the arithmetic and logical unit must do to find the answer, and send them back to us at the same frantic pace. Thus, for every classroom period of 45 minutes that we used the computer, less than a minute of the CPU time needed to be devoted to Lee High School. In a time-sharing computer, the memory is split up into many parts, each one devoted to one of the many users. The control unit shunts back and forth from one part of the memory to the other, and assigns the chores of the arithmetic and logical unit to each part in turn. The speed of this operation is so great that it is extremely uncommon that the user ever has to wait more than 5 seconds. Thus, although the use of the CPU costs $144 per hour, the net cost to the user is but a few dollars per hour.

Now that we saw how we use the computer, what could it do? The initial questions were often quite naive. "Will the computer tell my fortune?" was not an unusual question from the students. Most students felt that the computer was some omniscient genius with supernatural powers that was smarter than any of them and never could make mistakes. Usually, however there was a student in each class who had worked in a bank or office and had some degree of sophistication about the computer age. This student would explain to the rest of the class that the computer is really not very smart. In fact, it can only do what it is told to do. In short, "the computer is stupid." This soon became apparent to the students when they used the computer. For example, when someone made the usually harmless misspelling of the word "no" by spelling it with an N and a "zero," instead of an N and an "oh" the computer came back with "Sorry, I do not understand your statement.

Please retype it." Sometimes the computer would come up merely with an exasperated "what?". It soon became obvious, even to the dullest student, that here was a being whose intelligence was on a far lower order than his. How limited this intelligence is, we shall soon see.

Let us turn our attention first to the memory. The memory assigned to each user holds about as much information as a single, legal sized, single-spaced, narrow-margined typewritten page. One of the first things that the computer must do is assign to its memory the program, which is the list of instructions which we issue to it. This program tells the computer exactly what to do. There is one complication here. Because the computer is unbelievably stupid, the instructions must be equally specific. For example, suppose a secretary gave you a letter that she had typed and asked you what to do with it. You might say to her "you've misspelled the word ' teletype '". Look it up in the dictionary and correct it. Also instead of saying "digital, electronic computer", please change it to "electronic, digital computer". If you were dealing with a computer instead of a secretary your orders might have to go like this "go to line 13. Delete the characters in spaces 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23. In space 12 place a "t", in space 13 place a "y" in space 14 . . . , etc." Clearly, if any reasonable person had to work with a secretary who required such literal directions, she would soon be fired. The same is true for computers. No one has the patience to speak to a computer in its own, unbelievably simple-minded language, which has little resemblance either to English or the type of mathematical language that scientists, engineers, bank clerks or students are accustomed to using. For this reason, we wrote our program (list of instructions to the computer) in a language called BASIC, which was developed at Dartmouth College by Kemeny. BASIC is by no means a forbidding language. In fact, experience has shown that high school teachers can learn to program in BASIC after about 20 hours of a combination of classroom instruction and experience at the teletypewriter. The program "SLANG" is listed in the appendix at the end of this article.

The essential thing to note about the program is that, unless the user wishes to change it, it stays in the computer memory unchanged for the duration of the job. On the other hand, each job has a different set of data which are fed into the computer. For example, in a bank the amount of money for each check or deposit is a single datum. Handling all these numbers is called data processing. The computer program, which is made up by a computer programmer, may remain unchanged in the computer for a given bank for many years, even though each check and deposit is for a different amount. Likewise, to translate slang into English or vica versa, the program remains the same. Only the data were changed. The data, in this case, came in two parts. The first part consisted of a list of about 50 words in standard English and their slang equivalents. The second set of data consisted of sentences in either slang or standard English that the students wanted to be translated.

We have seen that the sum total of all the information we stored in the computer's brain consisted of not much more than one typewritten page, about 6,000 letters or about 1000 words of average length. This is not say that computers of larger memories are not available. For example the company that sold time to us had a larger computer which had a memory about twice as large. Even so, this is not an overly impressive amount of information to be stored in the brain of even the most ill paid and poorly educated clerk in the high school.

Slang and the Computer

We began with the idea of a program that would print out a Slang/Standard English Dictionary, translate (transpose would be the better word) from Slang into Standard English, or translate from Standard English to Slang. We thought of this program as a classroom "gimmick," a device to arouse and focus interest in two aggressive, intelligent, but quite underachieving classes of inner-city Freshmen. This general aim proved to be as testable as it seems; the gadget reigned over the classroom with an authority and attractiveness few lesson plans promise and fewer deliver. But the program proved to run more deeply through the lives of our students than we had supposed.

Mark Van Doren once said that "The best way to learn something is to teach it." The device of using underachievers as teachers of subordinate underachievers, usually younger students, has frequently been applied in inner-city education. It is an early form of affective education, a means of building ego, the sense of responsibility, and language skills at once. Generating a dictionary for our purposes seemed easy enough; we distributed three-by-five cards and asked our students to compose sentences using slang terms, indicating a standard English translation as well. The cards would then be collected, edited for redundancies, errors, and the requirement of a fifteen-letter limit per definition, and then fed to the computer as a list. In the actual classroom situation, however, we found that we were not writing a dictionary. As we simulated translations, struggled for definitions, and wrestled, as many of our students had all their lives wrestled, with the discomformation of slang and standard English, it became clear that we were creating a student.

After the initial period of open discussion of which the computer was de-romanticized from something on the order of an omniscient super-brain, and hence dubbed "stupid" on the first of many counts, we tried to present our proposed Slang Standard English Program to the class. The notion of translation proved, quite properly, to be somewhat difficult for the class. A small number of the students were taking a foreign language, and a larger number were bi-lingual naturally, for the most part in Spanish. We picked up a number of Puerto Rican slang terms and their Standard English equivalents. The class then practiced oral translations from one to the other, using the list or "dictionary" we had generated during our discussion. Our first difficulty arose when we tried the same format for the purposes of generating a Slang Standard English dictionary. Even our relatively elementary attempts at translation in this domain did not seem to work. Orville Barrow proposed the slang word "boss" and agreed that "nice" would do as its Standard English equivalent. But when his sentence, "I like her; she's pretty boss," was transmuted into "I like her; she's pretty nice," he refused to accept the translation. Time and again the stylistic function of the slang term, its aura of authenticity and inside-dopesterism, would be lost in translation, until it became clear that what our program was set against was a fragile, but essential, aspect of slang usage.

The computer was deemed "stupid" on a second count. Slang serves to identify the speaker as a member of an in-group; inner-city slang is highly developed, elusive, fast-changing, all in service of its hermetic, identification function. The class came to see that the computer was not going to loom as a member of the in-group speaking inner-city slang. Instead, "understanding" only the dictionary with which we informed it, the computer would be nothing more than a groping student of the relationship between inner-city Slang and Standard English.

Teaching the Computer

Although the instructions which went with the three-by-five cards we distributed at this point were to write a sentence in slang and then its Standard English translation, the cards came back in a variety of formats. Some students translated only the slang word; others omitted that because they were, quite rightly, at a loss for an adequate equivalent. But, cards in hand, we determined to teach the computer to "speak" inner-city slang to the best of our ability. The first step was to get into shape our cards, our list of equivalents and the first step toward the dictionary we were going to feed as data to the computer.

The cards varied greatly in freshness, authenticity, informativeness and degree of completion. We worked through them one-by-one in the classroom. They occasioned several asides into the nature of Standard English Grammer; several students had, for example, changed the spelling of syntax of their contextual sentence in order to give a greater authenticity to their use of their slang term. Thus, Frank Caraballo wrote, "Listen here comes the fuzz be cool and you don't know nothin." The urgent, unpunctuated rush of the sentence, his correct spelling of "don't" but incorrect omission of the final *g* from "nothing," as well as his slang words, were subjects for discussion. We admired the sentence as a recreation of dialect and for its drama, and complimented Frank on his ear. We translated fuzz as police and be cool as relax -- after considerable discussion. Not all of our output was as rich. Here is a sampling of it:

Look at that broad over there across the street.
She was looking up tight for you Jim.
I saw a boss man on the way home.
The boy I was talking to is a clean dude.
I am going to get stoned.
All the boys think Marie is a fox; do you?
I am going to be looking out of sight tonight.
They were a bunch of cut ups
he was dizzy
He's savage.
About three years ago there was a chick that thought she was boss.
She was diz from all the other broads.
We had to split.
The boy took a drag of the cigarette.

Many of our slang words, especially dude and cool, appeared more than once, often in variant usages. Refining our definitions, deciding upon alternative strategies to accomodate more than one aspect of a word's usage, and pursuing worthwhile and interesting asides as they arose, we arrived at a list which took only a few hours of work behind the scenes to become our first dictionary. See Appendix A. We fed our dictionary to the computer; rather, we fed our list of slang words and their Standard English equivalents to the computer, which had been programmed to alphabetize them into both a Slang-Standard English and a Standard English-Slang dictionary. It would doubtlessly be best to do at least part of this stage with the class, if possible; the phenomena of computer operation, especially of the alphabetization of our list into a Slang/Standard English, Standard English/Slang dictionary, are exciting and interesting. At a later stage we added to this dictionary in the classroom. Also at a later time we ran off copies of the dictionary for classroom use. But at this point our next task was to introduce the class to our computer's fund of knowledge: our dictionary. We did this by dictating the dictionary in class.

Testing Our Student

Each student in the class had to record his own copy of our dictionary. Part by part, the dictionary was put up on the blackboard as it was dictated to the class. Abstracted from their original context, the sentences the class had composed, our words occasioned much less discussion than they had at the outset. Nonethless, we had to justify many of our decisions in terms of limitations of space allowed by the program and limitations in the flexibility of our program. By a third count, the computer was "stupid." Nonetheless, each equivalency was entertained in turn. The class was impressed with the exacting simplemindedness of the computer. We had to be careful to stick to the dictionary if we wanted to avoid confusing the computer; for example, *cool* is translated as *great*, but *my cool* is translated as *my temper*, as in "That *jive turkey* made me *lose my cool*." We explained that, given that sentence to translate, the computer would not translate *jive turkey*. Neither *jive* nor *turkey* is in our dictionary. If we wanted the computer to deal with *jive* and *turkey*, we would have to arrive at equivalents in Standard English and add them to the program. What the computer would do is write, "That jive turkey made me lose my temper." That was interesting enough. The class wanted to see.

As a homework assignment, begun under supervision in class, the class was instructed to compose a dialogue, short paragraph, or series of sentences using the slang side of our dictionary; the next day, these compositions would be fed to the computer in class by students to see whether the computer could translate them into Standard English. The class was reminded to stick to the dictionary; no slang term, no matter how attractive, would be dealt with by the computer unless it were in our dictionary. Class response to the assignment was good, and assistance and support were directed to the slower starters using their original effort, the three-by-five card, as the germ for some kind of composition.

A good supply of usable *tests* came from the homework assignment. Most mingled slang and Standard English words with abandon, and required some disciplined editing before they could be submitted with best effect to the computer. Here are some representative essays:

> The tudge is groovy but is a cut up. She gets stoned every night and goes wild and she's stiff for days. Her dudes gigged and took a couple of drags of reefer. The fuzz raided the pad and busted the tudge.
>
> -Vivian Blue

A boy was walking down the street mugging with some dudes. They were a bunch of cut ups. A clean dude got in their way. He was savage, but they gave him action and split.

-Steve Shusky

A Cut Up

There was an out of sight chick but one thing about her was that she was dizzy. She was always laughing, screaming and always for a drag. She never bought her own smokes. Once she went to school stoned. Mira awful. She was an estudiante at a public high school but she got busted and had to split school for good.

-Karen DeSilva

About three years ago there was a chick that thought she was boss. She was diz from all the other broads. At a party she showed her friends how she could swing. She was wild. One night she started clowning, cutting up and she looked as if she was stoned. Finally she lost her cool and we had to split.

-Willie Kearney

The Scene

One night me and some dudes and a guy we called "Holmes" went out for some action. One of the boys we always had to tell him "be cool, be cool," because he always blew it whenever he saw a tudge go by. One of the dudes was called "Poka." He wore some bad rags! He was together!

Well, we got stiff and we were really mellow yellow. And guess who lost his cool when a broad passed by in a micro mini.

We split that scene and let him have all the action. When we asked him what happened he said "skip it."

So that was that.

-Elaine Peters

Suspense is the postponement of gratification. These compositions were fed to the computer by three to four earnest students typing collectively at once on one keyboard, making, remarkably, very few mistakes. At this point, the computer as a student was accepted as an entity in his own right, and his awkward translations proved to be as delightful as they were bungling. Typed back at about a hundred words a minute, they had a kind of authority, dimly suggesting the incredible speed that is the actual syntax of the computer. Slang words which the computer ignored were, of course, not in our dictionary, and were noted for entry at a later date. We took our translations and, as a classroom exercise, reworked several of them into standard English prose.

Mastering the Difference

Once an inner-city class has seen the slang they variously master for the changing and elusive language it is, standard English seems far more sure and tractable by comparison. Time and again, the Slang-Standard English Dictionary proved itself as ludicrous as it was interesting. We were entertained by its failures, and then moved beyond that entertainment both to conjectures as to how the dictionary might be improved, and to a more adequate "translation" attempt of our own. We corrected the computer's translation corporately, by rewriting it on the blackboard according to the largely consensual dictates of the class, and led by the teacher's standards for Standard English prose.

Take for example, the slang essay written by Willie Kearney, by use of the Dictionary. (Fig. 1) Fig. 1 shows the essay in its original, handwritten form: The class selected it for the computer, and programmedit in as data, following the simple artificialities of the program. They of course modified it somewhat in the doing, an almost invariable and enriching process whenever they had data in hand. The essay survived in this form: (Figure 2) The computer "translated" it into this: (Fig. 3) Had it been able to translate "broads," the computer would have met the class' acclaim; instead, it suffered the class' derision for being incapable of recognizing the plural form of a noun it knew in the singular. So, we rewrote the appropriate line of data using the singular form of the noun, and gave the computer another chance: (Fig. 4)

The computer rose to the occasion.

We matched the computer's final effort against the original slang essay. We all preferred the essay in its original slang form. The class set about bringing the translation up to standards by rewriting it on the blackboard, trying to find more adequate lexical and syntactical equivalents for the sense of the original. This challenge was good for a homework assignment and took two days. Pooling our efforts, we arrived at this essay:

> About three years ago there was a very attractive girl who knew it, too. She was a bit conceited. She was unique, different from all the other pretty girls. Once at a party she was showing her friends how she could dance. She was splendid. As it got later, she became more and more outgoing. She started to act foolishly and misbehave. She looked as if she were intoxicated. Finally, she lost her self-control and we had to leave. Now she seemed like a stupid girl to me for having gotten intoxicated and having made me lose my temper.

In its final form, our essay became:

> About three years ago I dated a very attractive girl who knew it, too. She was more than a bit conceited. She was different from the other pretty girls. Once at a party she began to show her friends how she could dance. She was splendid, but as it got later, she became more and more excited. She started to act foolishly, as though she were intoxicated. Finally, she lost her self-control and we had to leave. Now, having gotten intoxicated and having made me lose my temper, she seemed like a stupid kid.

Figure Captions

Figure 1- This is the essay, as it came from the student, in his own handwriting.

Figure 2- This is the printout of an actual run, as it was typed by the teletypewriter. The first line is a heading, which identifies the program, time and date. Then the computer asks the person at the typewriter what he or she wants. After the person types "English," the computer gives instructions for typing in the sentences in slang. The person enters the "data" and tells the computer to "run." The results are shown on Fig. 3. (The word "unit" refers to the time, in seconds, that the computer used to do its work.)

Figure 3- In less than a minute, the teletypewriter prints out this translation. The class is not satisfied with the translation.Good English should not have the word "Broads" in it. The person operating the teletypewriter types "yes" in answer to the question"Do you want to do more?"

Figure 4- This time, the class is more careful to put the word "Broad" in a form the dictionary will understand. Only a single line of the text has to be typed in, the line to be corrected. The computer remembers the rest. When told to "run" and print out "English," it types out the revised text in seconds.

Figure 5- Simplified flow chart of "Slang" to show the logic of the program.

Figure 6- "List" of "Slang." This is the actual program in BASIC, as fed into the computer.

Figure 7- Reconstruction of a run in the classroom. Langston Hughes' poem, "Motto", was fed into the computer in lines 2000-2006, which begin with the word "DATA." Before the computer could translate this poem into standard English, the pupils found they had to teach the backward computer a few words, which are given in the DATA in line 1017. Then the computer could spout forth with its translation.

ENGLISH-SLANG AND SLANG-ENGLISH DICTIONARY, AS PRINTED BY THE COMPUTER.

ENGLISH	SLANG
ACTING BADLY	MUGGING
ACT BADLY	MUG
ALL RIGHT	CLEAN
A TROUBLE MAKER	A CUT UP
ATTITUDE	TUDE
BEAUTIFUL	OUTASIGHT
CHICKEN	GALLINA
CLOSE TO	UP TIGHT FOR
CRAZY	ZONK
DANCE	SWING
DRUNK	STIFF
DUMB	ZULK
EXCITEMENT	ACTION
EXCUSE ME	HEY
FALL DOWN	CAERSE
FOOL	CLOWN
FOOLING	CLOWNING
FRIEND	DUDE
GET OUT	SCREW
GIRL	CHICK
GIRL	BABY
GOOD	BAD
GREAT	COOL
GREAT	FLY
HER TEMPER	HER COOL
HIGH	MELLOW YELLOW
HIS TEMPER	HIS COOL
INTOXICATED	STONED
LANDING	ATERRIZANDO
LEAVE	SPLIT
LOOK	MIRA
LOSE	BLOW
LOSES	BLOWS
LOST	BLEW
MISBEHAVE	CUT UP
MY TEMPER	MY COOL
NEVER MIND	SKIP IT
NICE	BOSS
NICE	KEEN
ODD	DIZ
PACE SETTER	SAVAGE
PRETTY GIRL	BROAD
RELAX	BE COOL
SIR	MAN
SMOKE	DRAG
SPLENDID	WILD
STUDENT	ESTUDIANTE
STUPID	DIZZY
STYLISH	GEAR
SUPERIOR	TUFF
TEENAGED GIRL	TUDGE
THEIR TEMPER	THEIR COOL
UP TO DATE	GROOVY
WOMAN	FOX
WONDERFUL	OUT OF SIGHT
YOUR TEMPER	YOUR COOL

SLANG	ENGLISH
ACTION	EXCITEMENT
A CUT UP	A TROUBLE MAKER
ATERRIZANDO	LANDING
BABY	GIRL
BAD	GOOD
BE COOL	RELAX
BLEW	LOST
BLOW	LOSE
BLOWS	LOSES
BOSS	NICE
BROAD	PRETTY GIRL
CAERSE	FALL DOWN
CHICK	GIRL
CLEAN	ALL RIGHT
CLOWN	FOOL
CLOWNING	FOOLING
COOL	GREAT
CUT UP	MISBEHAVE
DIZ	ODD
DIZZY	STUPID
DRAG	SMOKE
DUDE	FRIEND
ESTUDIANTE	STUDENT
FLY	GREAT
FOX	WOMAN
GALLINA	CHICKEN
GEAR	STYLISH
GROOVY	UP TO DATE
HER COOL	HER TEMPER
HEY	EXCUSE ME
HIS COOL	HIS TEMPER
KEEN	NICE
MAN	SIR
MELLOW YELLOW	HIGH
MIRA	LOOK
MUG	ACT BADLY
MUGGING	ACTING BADLY
MY COOL	MY TEMPER
OUTASIGHT	BEAUTIFUL
OUT OF SIGHT	WONDERFUL
SAVAGE	PACE SETTER
SCREW	GET OUT
SKIP IT	NEVER MIND
SPLIT	LEAVE
STIFF	DRUNK
STONED	INTOXICATED
SWING	DANCE
THEIR COOL	THEIR TEMPER
TUDE	ATTITUDE
TUDGE	TEENAGED GIRL
TUFF	SUPERIOR
UP TIGHT FOR	CLOSE TO
WILD	SPLENDID
YOUR COOL	YOUR TEMPER
ZONK	CRAZY
ZULK	DUMB

Appendix B. The Program

Figure 5 shows a "flow chart," which is a diagram to show how the program is organized. Figure 6 is a copy of the program. The computer reads one line at a time and follows the instructions before going on to the next line.

Lines 10 and 20 tell the computer to print the explanation of what it can do. (Lines 30-80 are remarks which explain how to use the program and have no effect on the computer.) After the computer reaches line 99, it jumps to line 150 and continues to type its explanation. Line 195 reserves space in the computer memory for 100 English words (A$) and 100 slang words or phrases (B$). Line 200 tells the computer to wait for your answer to the question in line 190.

There are three possibilities, which depend on the answer. If the answer is "dictionary," the computer jumps to line 260. The program from line 260 to line 360 arranges the English words in alphabetical order and is careful to sort the corresponding Slang terms. Lines 370-390 print out the English-Slang dictionary. Lines 400-498 repeat the whole process for the Slang-English dictionary. The results of this clerical work, shown in appendix A, take less than a minute of the computer's time. If the answer is "SLANG" or "ENGLISH," the computer goes to lines 860-873, which are instructions for putting in the Slang or English sentences. The computer "run" ends at line 873.

Typing "1 GOTO 100," tells the computer to start at line 100. This skips the long introduction and proceeds to line 190, where it asks what is wanted. This time, the answer will be "Slang" or "English." However, the presence of the instructions in lines 100 and 208 now make the computer jump to lines 500-760. It reads the words in the data one by one, searches through the dictionary for the word, and translates it. If it fails to find the word in the dictionary, the word stands. When it finishes the list it encounters the word "Z1234" in line 1999, which acts as a signal to follow the instruction in line 510 to proceed to line 800. The computer asks if you want to do more translations. If the answer is "no," the computer stops. If it gets a "yes" answer, it gives a brief reminder (lines 842-843) and stops the run.

A few examples of several "branchs" in this program are given in figures 2-4 and in appendix A.

FIGURE 1

Willie

Kearney

About three years ago there was a chick that thought she was boss. She was dig from all the other go broads. At a party she showed her her friends how she could ~~too~~ swing. She was wild. One night she started clowning, cutting up and she looked as if she was stoned. finally she lost her cool and we had to split.

STUDENT ESSAY

Fig. 2

```
SLANG      13:36   027  03/16/70

THIS IS A PROGRAM TO TRANSLATE ENGLISH INTO SLANG OR SLANG
INTO ENGLISH.
PLEASE TELL ME WHAT YOU WANT ME TO DO.
TYPE 'ENGLISH' IF YOU WANT ME TO TRANSLATE SLANG INTO ENGLISH.
TYPE 'SLANG' IF YOU WANT ENGLISH TRANSLATED INTO SLANG.
TYPE 'DICTIONARY' IF YOU WANT A SLANG-ENGLISH DICTIONARY.
WHAT DO YOU WANT, SLANG, ENGLISH OR DICTIONARY? ENGLISH
IN LINES 2000, 2001, ETC. ENTER 'DATA' AND TYPE IN THE SENTENCES
THAT YOU WANT. WRITE THE CORRECT WORD, TYPE A COMMA,
AND THEN THE NEXT WORD EXACTLY AS IT APPEARS IN THE DICTION-
ARY. LEAVE NO SPACES BETWEEN WORDS AND COMMAS. (TO PUT IN
PUNCTUATION MARKS, AS PERIODS OR COMMAS, PUT QUOTATION MARKS
AROUND THEM.)

  IN LINE 1, TYPE GOTO 100
  THEN TYPE  RUN
FOR EXAMPLE, TO PRINT 'LET'S GO MAN.', TYPE:
2000 DATA LET'S,GO,MAN,'.'
1 GOTO 100
RUN




USED    12.17 UNITS
TAPE
READY.

2000 DATA       ABOUT,THREE,YEARS,AGO,THERE,WAS,A,BOSS,CHICK,Y←
2001 DATA  THAT,THOUGHT,SHE,WAS,BOSS,OUT OF SIGHT,".",SHE,WAS,
2002 DATA  DIZ,FROM,ALL,THE,OTHER,BROADS,".",AT,A,PARTY,SHE,SHOWED,
2003 DATA  HER,FRIENDS,HOW,SHE,COULD,SWING,".",SHE,WAS,WILD,".",
2004 DATA  ONE,NIGHT,SHE,STARTED,CLOWNING,CUTTING UP,".",SHE,CUT UP,
2005 DATA  BAD,".",SHE,LOOKED,AS,IF,SHE,WAS,STONED,".",FINALLY,SHE,
2006 DATA  LOST,HER,COOL,AND,WE,HAD,TO,SPLIT,".",THAT,DIZZY,FOX,
2007 DATA  GOT,MELLOW YELLOW,AND,MADE,ME,BLOW,MY COOL,CLEAN,".",
1 GO TO 100
RUN
```

Fig.

```
SLANG      13:40   027  03/16/70

WHAT DO YOU WANT, SLANG, ENGLISH OR DICTIONARY? ENGLISH
ABOUT THREE YEARS AGO THERE WAS A
NICE GIRL THAT THOUGHT SHE WAS NICE
WONDERFUL .
SHE WAS ODD FROM ALL THE OTHER
BROADS .
AT A PARTY SHE SHOWED HER FRIENDS
HOW SHE COULD DANCE .
SHE WAS SPLENDID .
ONE NIGHT SHE STARTED FOOLING CUTTING UP .
SHE MISBEHAVE GOOD .
SHE LOOKED AS IF SHE WAS INTOXICATED .
FINALLY SHE LOST HER GREAT AND WE
HAD TO LEAVE .
THAT STUPID WOMAN GOT HIGH AND MADE
ME LOSE MY TEMPER ALL RIGHT .

DO YOU WANT TO DO MORE (TYPE YES OR NO)? YES
FOLLOW THE INSTRUCTIONS I GAVE YOU BEFORE ABOUT TYPING IN
SENTENCES.




USED    27.17 UNITS
2002 DATA DIZ,FROM,EVERY,OTHER,BROAD,".",AT,A,PARTY,SHE,SHOWED,
RUN
```

Fig.

```
SLANG      13:43   027  03/16/70

WHAT DO YOU WANT, SLANG, ENGLISH OR DICTIONARY? ENGLISH
ABOUT THREE YEARS AGO THERE WAS A
NICE GIRL THAT THOUGHT SHE WAS NICE
WONDERFUL .
SHE WAS ODD FROM EVERY OTHER PRETTY GIRL .
AT A PARTY SHE SHOWED HER FRIENDS
HOW SHE COULD DANCE .
SHE WAS SPLENDID .
ONE NIGHT SHE STARTED FOOLING CUTTING UP .
SHE MISBEHAVE GOOD .
SHE LOOKED AS IF SHE WAS INTOXICATED .
FINALLY SHE LOST HER GREAT AND WE
HAD TO LEAVE .
THAT STUPID WOMAN GOT HIGH AND MADE
ME LOSE MY TEMPER ALL RIGHT .

DO YOU WANT TO DO MORE (TYPE YES OR NO)? NO
IT'S BEEN A PLEASURE TO SERVE YOU. COME AGAIN.



USED    30.67 UNITS
```

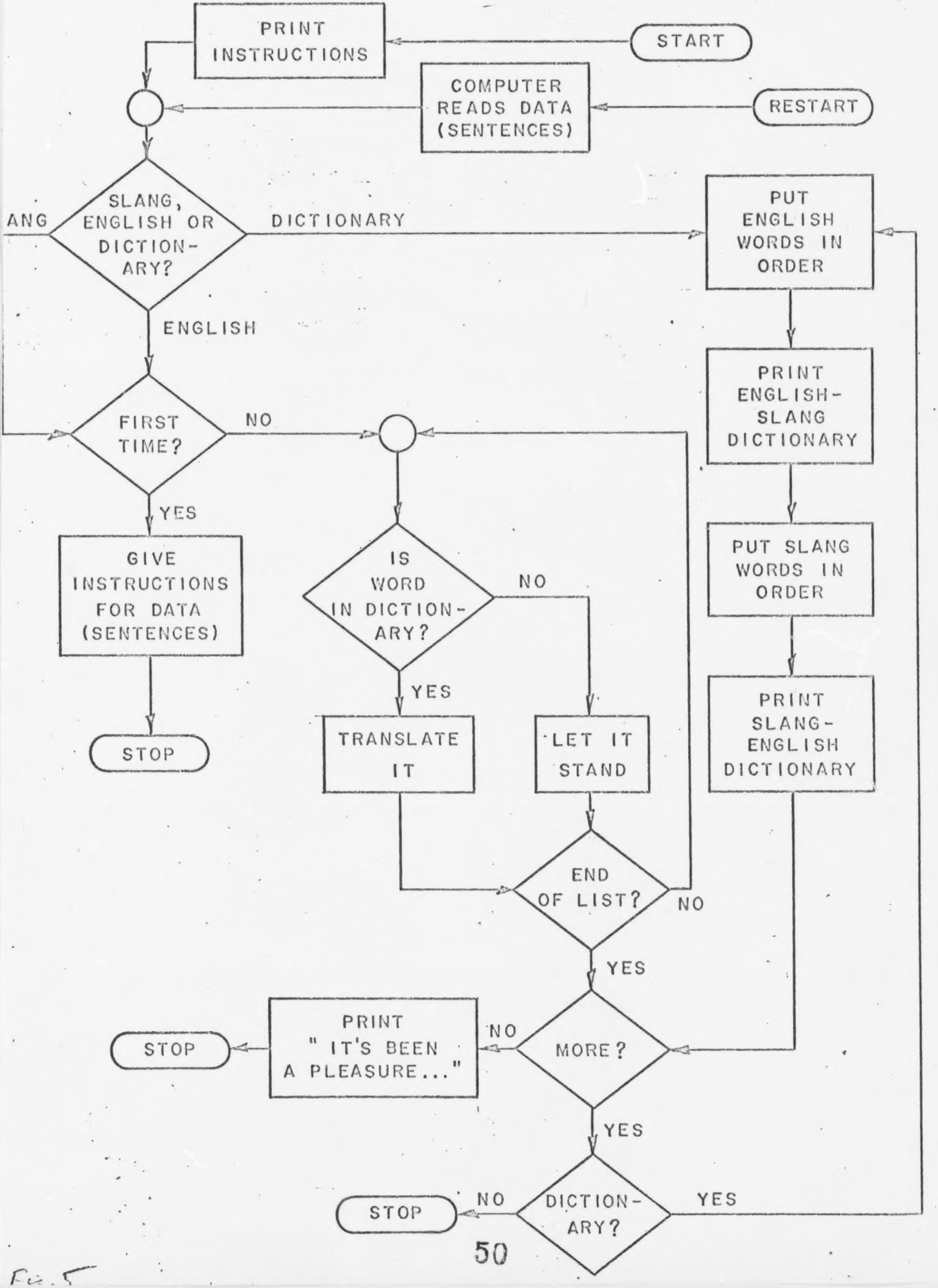

START
PRINT INSTRUCTIONS
RESTART
COMPUTER READS DATA (SENTENCES)
SLANG, ENGLISH OR DICTIONARY?
ANG
DICTIONARY
ENGLISH
PUT ENGLISH WORDS IN ORDER
PRINT ENGLISH-SLANG DICTIONARY
PUT SLANG WORDS IN ORDER
PRINT SLANG-ENGLISH DICTIONARY
FIRST TIME?
NO
YES
GIVE INSTRUCTIONS FOR DATA (SENTENCES)
STOP
IS WORD IN DICTIONARY?
NO
YES
TRANSLATE IT
LET IT STAND
END OF LIST?
NO
YES
MORE?
NO
PRINT " IT'S BEEN A PLEASURE..."
STOP
YES
DICTIONARY?
NO
STOP
YES

Fig. 5

FIG 6

```
10 PRINT"THIS IS A PROGRAM TO TRANSLATE ENGLISH INTO SLANG OR SLANG"
20 PRINT"INTO ENGLISH."
30 REM LINES 1000-1998 ARE RESERVED FOR THE ENGLISH-SLANG WORDS AND THER
40 REM EQUIVALENTS. IN LINE 1000, ENTER THE WORD 'DATA',TYPE THE ENGLISH
50 REM WORD, A COMMA, THE SLANG WORD, A COMMA, AND SO FORTH. LEAVE NO
60 REM SPACES BETWEEN WORDS AND COMMAS.FOR EXAMPLE:"
70 REM 1000 DATA LEAVE,SPLIT,ATTITUDE,'TUDE,GO AWAY,GET LOST
80 REM YOU MAY USE UP TO 100 WORDS.
98 LET X = 0
99 GOTO150
100 LET X = 1
110 GOTO 190
150 PRINT"PLEASE TELL ME WHAT YOU WANT ME TO DO."
160 PRINT"TYPE 'ENGLISH' IF YOU WANT ME TO TRANSLATE SLANG INTO ENGLISH"
170 PRINT"TYPE 'SLANG' IF YOU WANT ENGLISH TRANSLATED INTO SLANG."
180 PRINT"TYPE 'DICTIONARY' IF YOU WANT A SLANG-ENGLISH DICTIONARY."
190 PRINT"WHAT DO YOU WANT, SLANG, ENGLISH OR DICTIONARY";
195 DIM A$(100),B$(100)
200 INPUT D$
201 FOR J =1 TO 300
202 READ A$(J), B$(J)
203 IF B$(J) = "Z1234" THEN 207
204 LET N = J
205 NEXT J
207 IF D$ = "DICTIONARY" THEN 260
208 IF X = 1 THEN 500
210 IF D$ = "SLANG" THEN 860
220 IF D$ = "ENGLISH" THEN 860
240 GOTO 190
250 REM LINES 250-499 ARE FOR MAKING UP THE DICTIONARY.
260 PRINT
261 PRINT
262 PRINT "ENGLISH","SLANG"
263 PRINT
265 FOR I = 1 TO N
270       FOR J = 1 TO N-I
280       LET X$ = A$(J)
290       LET Y$ = A$(J+1)
300       IF X$ <= Y$ THEN 350
310       LET A$(J) = Y$
320       LET A$(J+1) = X$
330       LET T$ = B$(J)
340       LET B$(J) = B$(J+1)
345       LET B$(J+1) = T$
350       NEXT J
360 NEXT I
370 FOR I = 1 TO N
380 PRINT A$(I), B$(I)
390 NEXT I
```

```
380 PRINT A$(I), B$(I)
390 NEXT I
400 FOR I = 1 TO N
410      FOR J = 1 TO N-I
420      LET X$ = B$(J)
430      LET Y$ = B$(J+1)
440      IF X$ <= Y$ THEN 490
450      LET B$(J) = Y$
460      LET B$(J+1) = X$
470      LET T$ = A$(J)
480      LET A$(J) = A$(J+1)
485       LET A$(J+1) = T$
490      NEXT J
491 NEXT I
492 PRINT
493 PRINT
494 PRINT"SLANG","ENGLISH"
495 PRINT
496 FOR I=1 TO N
497 PRINT B$(I),A$(I)
498 NEXT I
499 GOTO 800
500 LET K = 0
501 READ C$
502 IF C$ = "." THEN 508
503 IF C$ = "," THEN 508
504 IF K<7 THEN 510
505 PRINT
506 LET K = 0
507 GOTO 510
508 PRINT C$
509 GOTO 500
510 IF C$ = "Z4321" THEN 800
515 IF D$ = "SLANG" THEN 540
520 IF D$ = "ENGLISH" THEN 530
530 GOSUB 600
532 GOTO 501
540 GOSUB 700
542 GOTO 501
600 FOR I = 1 TO N
610 IF C$ = B$(I) THEN 650
620 NEXT I
630 PRINT C$;" ";
640 GOTO 655
650 PRINT A$(I);" ";
655 LET K = K+1
660 RETURN
700 FOR I = 1 TO N
710 IF C$ = A$(I) THEN 750
720 NEXT I
730 PRINT C$;" ";
740 GOTO 755
750 PRINT B$(I);" ";
755 LET K = K+1
760 RETURN
800 PRINT
801 PRINT"DO YOU WANT TO DO MORE (TYPE YES OR NO)";
810 INPUT E$
820 IF E$ = "YES" THEN 841
830 IF E$ = "NO" THEN 990
840 GOTO 800
841 IF D$ = "DICTIONARY" THEN 845
```

```
720 NEXT I
730 PRINT C$;" ";
740 GOTO 755
750 PRINT B$(I);" ";
755 LET K = K+1
760 RETURN
800 PRINT
801 PRINT"DO YOU WANT TO DO MORE (TYPE YES OR NO)";
810 INPUT E$
820 IF E$ = "YES" THEN 841
830 IF E$ = "NO" THEN 990
840 GOTO 800
841 IF D$ = "DICTIONARY" THEN 845
842 PRINT "FOLLOW THE INSTRUCTIONS I GAVE YOU BEFORE ABOUT TYPING IN"
843 PRINT "SENTENCES."
844 STOP
845 RESTORE
846 GOTO 150
860 PRINT"IN LINES 2000,2001,ETC,ENTER 'DATA' AND TYPE IN THE SENTENCES"
861 PRINT"THAT YOU WANT. WRITE THE CORRECT WORD, TYPE A COMMA,"
862 PRINT"AND THEN THE NEXT WORD EXACTLY AS IT APPEARS IN THE DICTION-"
863 PRINT"ARY. LEAVE NO SPACES BETWEEN WORDS AND COMMAS. (TO PUT IN "
864 PRINT"PUNCTUATION MARKS, AS PERIODS OR COMMAS, PUT QUOTATION MARKS"
865 PRINT"AROUND THEM.)"
866 PRINT
867 PRINT"  IN LINE 1, TYPE GOTO 100"
868 PRINT"  THEN TYPE  RUN"
869 PRINT"FOR EXAMPLE, TO PRINT 'LET'S GO MAN.', TYPE:"
870 PRINT"2000 DATA LET'S,GO,MAN,'.'"
871 PRINT"1 GOTO 100"
872 PRINT"RUN"
873 STOP
990 PRINT"IT'S BEEN A PLEASURE TO SERVE YOU. COME AGAIN."
1000 DATA  ODD,DIZ,DRUNK,STIFF,SUPERIOR,TUFF,GREAT,COOL,NICE,BOSS
1001 DATA  STYLISH,GEAR,"PACE SETTER",SAVAGE,"UP TO DATE",GROOVY
1002 DATA  "CLOSE TO","UP TIGHT FOR",BEAUTIFUL,OUTASIGHT,"YOUR TEMPER"
1003 DATA  "YOUR COOL","MY TEMPER","MY COOL","HIS TEMPER","HIS COOL"
1004 DATA  "HER TEMPER","HER COOL","THEIR TEMPER","THEIR COOL"
1005 DATA  DANCE,SWING,LOST,BLEW,LOSE,BLOW,LOSES,BLOWS,GIRL,CHICK
1006 DATA  GREAT,FLY,FRIEND,DUDE,EXCITEMENT,ACTION,WOMAN,FOX,NICE
1007 DATA  KEEN,MISBEHAVE,"CUT UP","A TROUBLE MAKER","A CUT UP"
1008 DATA  RELAX,"BE COOL",SIR,MAN,"EXCUSE ME",HEY,LOOK,MIRA
1009 DATA  "NEVER MIND","SKIP IT","TEENAGED GIRL",TUDGE
1010 DATA  "FALL DOWN",CAERSE,INTOXICATED,STONED,"GET OUT",SCREW
1011 DATA  LEAVE,SPLIT,ATTITUDE,TUDE,HIGH,"MELLOW YELLOW",SPLENDID
1012 DATA  WILD,"PRETTY GIRL",BROAD,GIRL,BABY,STUPID,DIZZY,SMOKE,DRAG
1013 DATA  CRAZY,ZONK,DUMB,ZULK,CHICKEN,GALLINA,STUDENT,ESTUDIANTE
1014 DATA  "ACT BADLY",MUG,FOOL,CLOWN,FOOLING,CLOWNING,"ACTING BADLY"
1015 DATA  MUGGING,GOOD,BAD,"ALL RIGHT",CLEAN,WONDERFUL,"OUT OF SIGHT"
1016 DATA  LANDING,ATERRIZANDO
1999 DATA A,Z1234
9998 DATA Z4321
9999 END
```

```
SLANG     15:22  27 WED 06/24/70

THIS IS A PROGRAM TO TRANSLATE ENGLISH INTO SLANG OR SLANG
INTO ENGLISH.
PLEASE TELL ME WHAT YOU W0NT ME TO DO.
TYPE 'ENGLISH' IF YOU WANT ME TO TRANSLATE SLANG INTO ENGLISH.
TYPE 'SLANG' IF YOU WANT ENGLISH TRANSLATED INTO SLANG.
TYPE 'DICTIONARY' IF YOU WANT A SLANG-ENGLISH DICTIONARY.
WHAT DO YOU WANT, SLANG, ENGLISH OR DICTIONARY? ENGLISH
IN LINES 2000, 2001, ETC. ENTER 'DATA' AND TYPE IN THE SENTEN-
CES THAT YOU WANT. WRITE THE CORRECT WORD, TYPE A COMMA,
AND THEN THE NEXT WORD EXACTLY AS IT APPEARS IN THE DICTION-
ARY. LEAVE NO SPACES BETWEEN WORDS AND COMMAS. (TO PUT IN
PUNCTUATION MARKS, AS PERIODS OR COMMAS, PUT QUOTATION MARKS
AROUND THEM.)

  IN LINE 1, TYPE GOTO 100
  THEN TYPE  RUN
FOR EXAMPLE, TO PRINT 'LET'S GO MAN.', TYPE:
2000 DATA LET'S,GO,MAN,'.'
1 GOTO 100
RUN




USED    17.17 UNITS.
TAPE
READY.


1017 DATA UNDERSTAND,DIG,UNDERSTOOD,DUG,TALK,JIVE.
2000 DATA I,PLAY,IT,COOL
2001 DATA AND,DIG,ALL,JIVE,","
2002 DATA THAT'S,THE,REASON
2003 DATA I,STAY,ALIVE,"."
2004 DATA MY,MOTTO,AS,I,LIVE,AND,LEARN
2005 DATA IS,DIG,AND,BE,DUG,
2006 DATA IN,RETURN,"."
1 GOTO 100
KEY
READY.

RUN




SLANG     15:26   27 WED 06/24/70

WHAT DO YOU WANT, SLANG, ENGLISH OR DICTIONARY? ENGLISH
I PLAY IT GREAT AND UNDERSTAND ALL
TALK ,
THAT'S THE REASON I STAY ALIVE .
MY MOTTO AS I LIVE AND LEARN
IS UNDERSTAND AND BE UNDERSTOOD IN RETURN .

DO YOU WANT TO DO MORE (TYPE YES OR NO)? NO
IT'S BEEN A PLEASURE TO SERVE YOU. COME AGAIN.
```

Fig. 7.

ED057597

Revised 6/16/69

INDEX TO THE HANDBOOK OF THE COURSE IN BASIC PROGRAMMING

Section I - Introductory Material

- 1 - Index to Handbook
- 2 - Announcement Letter
- 3 - Instructions for the course - Read these before you start!
- 4 - Model 33 Teletypewriter keyboard
- 5 - How To Write A Program
- 6 - Teacher's Guide for Classroom Demonstration Of Computer

Section II - Assignments

- Lesson I - Getting Started - Simple Printing and Arithmetic
- Lesson II - Input - Getting Data into the Computer
- Lesson III - Branches and Loops - Preparing a Bank Statement
- Lesson IV - Lists and Tables - Data Files - Filing with the Computer
- Lesson V - Subroutines, Functions, Defined Functions
- Lesson VI - Matrices - Their Use in DATA Processing. Matrix Algebra
- Lesson VII - Advanced BASIC - string manipulations - Editing commands. Mark II BASIC

(At present, sections VI, VII are incomplete - material will be added as it is made ready.)

Section III - Demonstration Program Library - (Alphabetical Listing - This library will be added to from time to time.) Please note that this material can be used freely but rights for reproduction are reserved.

00944

YALE UNIVERSITY
Office of Teacher Training

June 21, 1969

Dear master and practice teachers:

During the summer of 1969, the Yale Office of Teacher Training will sponsor a program to acquaint M.A.T. candidates with the use of the computer as an aid to instruction in mathematics, science and to a lesser extent, other subjects.

These activities will involve the use of two, portable terminals which are connected to a remote computer. The following rooms will permit use of both terminals simultaneously: 3, 8, 10, ~~[illegible]~~.

ACTIVITIES

1. Demonstration - There will be a demonstration of the computer in room 3, at 3:30 p.m., Monday, June 23.

2. Classroom Activities - You can demonstrate or use the computer in your classroom as a part of your regular teaching activities, even if you have no experience with computers. Please see me. I shall be at the high school from 8:30 - 5:30 on the days of the computer laboratory. On off days, I shall be at my office, Gibbs 323, phone 787-3131, ext. 2366.

3. Computer Laboratory - All teachers of mathematics and science will take the computer laboratory as part of their methods courses. The schedule is as follows:

	MATHEMATICS TEACHERS	SCIENCE TEACHERS
Demonstration	Monday, June 23	Monday June 23
Laboratory	Wednesday, June 25 Friday, June 27 Tuesday, July 1 Thursday, July 3 Monday, July 7 Wednesday, July 9 Friday, July 11 Tuesday, July 15 Thursday, July 17	Monday, July 21 Wednesday, July 23 Friday, July 25 Tuesday, July 29 Thursday, July 31

Time: 3:30 - 5:30 p.m.
Place: Room 3 (West Campus)

Some of you will be beginners; others will have considerable experience at programming. For this reason, the laboratory will be run on an informal basis, with each of you setting his own pace. I hope the experienced members of the laboratory will help the beginners. Most of all, I hope you all enjoy the laboratory and find it of value to you.

Sincerely yours,
William Lichten
William Lichten
Professor of Physics

Section I

3 - Instructions for the Course in BASIC Programming

1.- Use the Handbook. - This handbook has the materials you need to take the course. In addition, you will need copies of the BASIC manuals distributed by General Electric.

2 - Familiarize yourself with the material in this manual. Use the index and check to see if all materials are present. From time to time, you'll receive new or corrected pages.

The Model 33 Teletypewriter Keyboard sheet will be useful when you use the teletypewriter. Many, if not most, TTY's have keyboards which do not type the standard BASIC character set. This won't matter to the computer, if you hit the right keys. This sheet will help you.

The G. E. Manuals

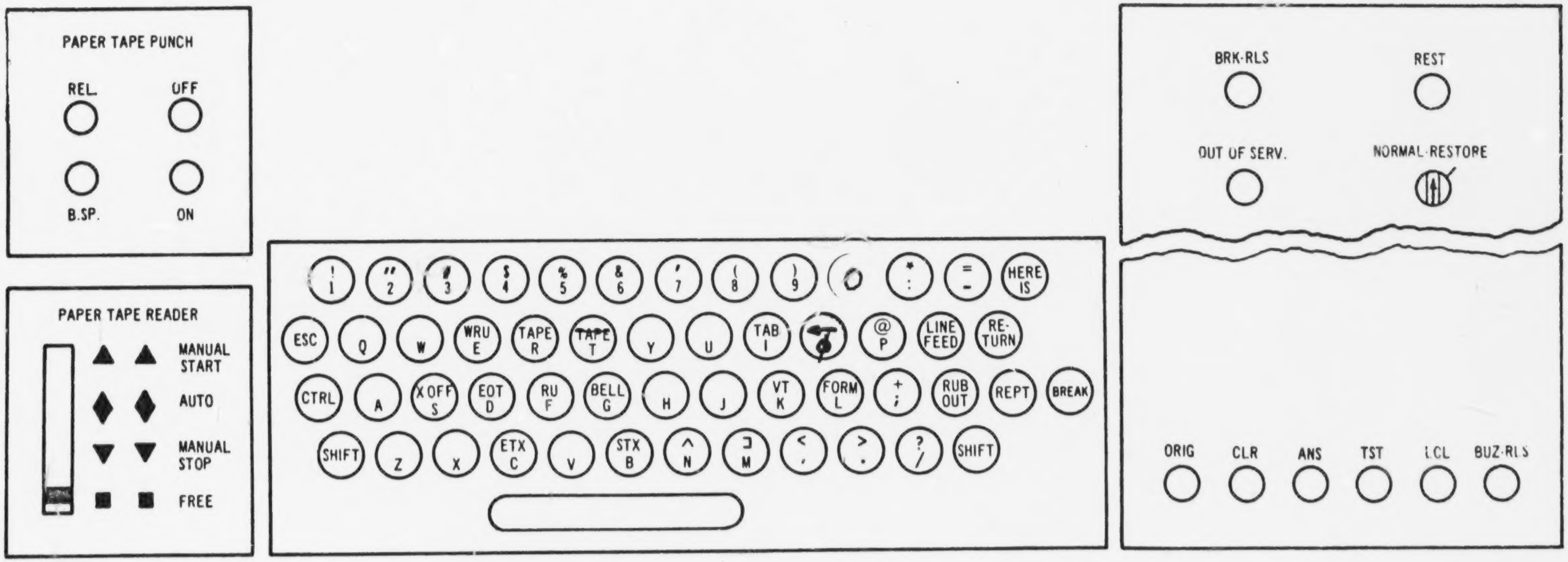
These are listed below more or less in the order that you learn BASIC.

1 - Introduction to BASIC Programming - Learner's Guide
2 - BASIC Language - Reference Manual
3 - Command System
4 - Mark II Reference Manual*
5 - Editing Commands

*(The Mark II computer is another GE computer that can be used. It is much larger and faster than Mark I. The BASIC language and command system are almost exactly the same as Mark I. The only impediment to using Mark II is that it is a long distance call to Bridgeport ($4.00 per hour). If and when local service to Mark II becomes available, we shall use it exclusively.)

MODEL 33 TELETYPEWRITER

The principal parts of the teletypewriter are the Control Unit, Keyboard, Paper Tape Punch and Paper Tape Reader.


PAPER TAPE PUNCH
REL.
OFF
B.SP.
ON
PAPER TAPE READER
MANUAL START
AUTO
MANUAL STOP
FREE
ESC
CTRL
SHIFT
WRU E
TAPE R
TAPE T
TAB I
LINE FEED
RE-TURN
HERE IS
X OFF S
EOT D
RU F
BELL G
VT K
FORM L
RUB OUT
REPT
BREAK
ETX C
STX B
BRK-RLS
REST
OUT OF SERV.
NORMAL-RESTORE
ORIG
CLR
ANS
TST
LCL
BUZ-RLS

Model 33 Keyboard and Controls

How to Write a Program

A. Before you come in, have your program written on paper.

B. Punch your program on tape, off-line (on "local" use). (See "Learner's Guide" for instructions.)

C. Feed your program in. (See bottom of p. 20 of "Learner's Guide" for instructions.)

D. "De-bug" your program. Usually the computer gives you an error message which immediately tells you what to do. You correct your program at the terminal, and run it off.

However, if you can't find your bugs right away, SAVE your program (see below), find what's wrong, and come back to correct it. If your corrections are more than a line or two, use tape to feed them in.

E. Saving your program.

1. While you are at the terminal, the program you are using will be saved and kept in correct order until you start working on another program, or sign off. For example, any of the commands, NEW, OLD, CAT, BYE will lose your program.
2. To save it for a short time, type "SAVE". Store only one program at a time. Store only the program you are using, or are writing and editing. All other programs should be stored on tape. To protect your saved program, use your initials for the first three letters of the name. John Harold Smith's program to add is named JHSADD
3. Don't store programs for long periods. Unless you plan to use a program within a few days, LISTNH it on tape, and UNSAVE it. (See p. 19 of "Learner's Guide" - section

titled "In the Time-Sharing Mode")

II. The Panic Button - Keep this with you while you are learning. This guide will get you out of trouble.

A. The STOP command. To get out while the computer is running (but not typing) type "STOP" or "S" and "return".

B. The "BREAK" command - This command interrupts running even when the computer is typing. Hold down the "break" key for about a second. To get back on-line, press the break-release button.

C. Ending the Conversation - If the computer is idle (not running) you can type "GOODBYE", or "BYE". When you do this, you disconnect the terminal from the phone line. This should not be necessary, as buttons A or B should always work. You'll lose your program, unless you SAVE it or LIST it on tape.

D. The CLR Button - This is the real panic button. Push it, and you disconnect the phone line. This stops everything. It has all the faults of C. In addition, you won't get a statement of your terminal connect time.

E. Correcting Mistakes -

1. On local mode (while tape-punching) - If you have a mistake, backspace the tape punch and press the "rubout" key. This will wipe your error out. You can backspace as many spaces as you want.

2. On local, or on line. You also can correct a mistake by pressing the "← " key. You can correct six mistakes by pressing "← " six times. This control simply

"rubs out" the mistake in the computer.

You can correct an entire line by typing its line number again and retyping it. If you just type the line number, you can delete the line. You also can delete, renumber, or move lines around with EDIT commands. (See "Command System".)

You can delete an entire line before you have gone on to the next line by pressing the "CONTROL" and "X" keys at the same time.

TIME IS MONEY. DON'T WASTE IT!

COMMON PROBLEMS:

A. If teletypewriter won't print on-line, check to see if there is a duplex switch - it should be on "HALF DUX" (up position)

B. If the tape reader won't work, check paper roll for being too low. Even if the "BUZ-RLS" switch is down, tape punch sometimes won't work. If so, replace paper roll.

C. If tape prints garbage on-line, it's usually because you forgot to type "TAPE". If it still types garbage, stop and run it through a second time. Sometimes, when garbage is printed, computer has the program correct.

Teacher's Guide for Classroom Demonstration of the Computer

1.- Reserve the computer and the classroom for the period you wish. The following rooms have telephone jacks and can be used:

Lee High School*:
South House 1214
West House 2214
North House 3104,3208, 3211
East House 4101, 4153, 4205 (Math department office), 4214

Hillhouse High School*: B-302 (Math department office); A-301 (physics workroom); B-317, B-314, B-313, B-312 (Math classrooms); Mrs. Hogen's office and Library stacks; A-215 (History classroom); Mrs. Green's office (2nd floor).

Wilbur Cross High School*: B-302 (Mathematics departmental office), B-310, B-318, A208, A301-A305 (physics storeroom).

North Haven High School*: 3,8,10.

*Often it is possible to use a classroom adjoining one of these rooms, by use of the long extension telephone cord that is with each terminal. (Length of extension cord + teletype cord is 50 feet)

2.- Make sure the demonstration program you want is saved in the computer for instant access.

3.- Have several copies of "You/and the Computer" on hand to show the students.

4.- Prepare the teacher to work effectively with the program. Most teachers like to go over the material the day before the demonstration.

5.- Give a short talk (15 - 20 minutes) about the computer and how it works.

6.- Seat the students up close to the terminal and let them operate it after you have dialed the computer and put in the user number. Have the students take turns sitting close to the terminal. (Not all the class can sit close enough to the teletypewriter to read the printing.)

The material given below covers what has been presented in past demonstrations. There is more material than can be covered in half a class room period. You will have to select the material which is best suited to your class.

How We Bring the Computer to You

Is this a computer that we wheeled into the classroom? No. This is not a computer. It is a teletype machine. It wouldn't be practical to bring a computer into the classroom. Page 18 of the booklet, "You/and the Computer" shows a picture of a modern, giant, electronic digital computer. As you can see from this picture, if we were to bring this computer into the classroom we would have to move the class out to make space for all the equipment. It is true that there are miniature electronic computers that are small enough to bring into this classroom. However they are not big enough to do all the things that we can do in this demonstration. That is why we like to use a giant computer.

Not only is this computer much too large to bring into this classroom, but it is much too expensive. The computer we normally use (a General Electric "Mark I" - the G.E.-265 time sharing computer.) costs approximately $500,000, about half a million dollars. This is too expensive for the New Haven Public Schools to buy. "Well" you might say, "even if it's too expensive to buy you can always rent one." The rental for this computer is about $16,000 per month. Even if we could only rent the computer for an hour it would cost $144. As you can see, this would still be much too expensive for us to use in the school.

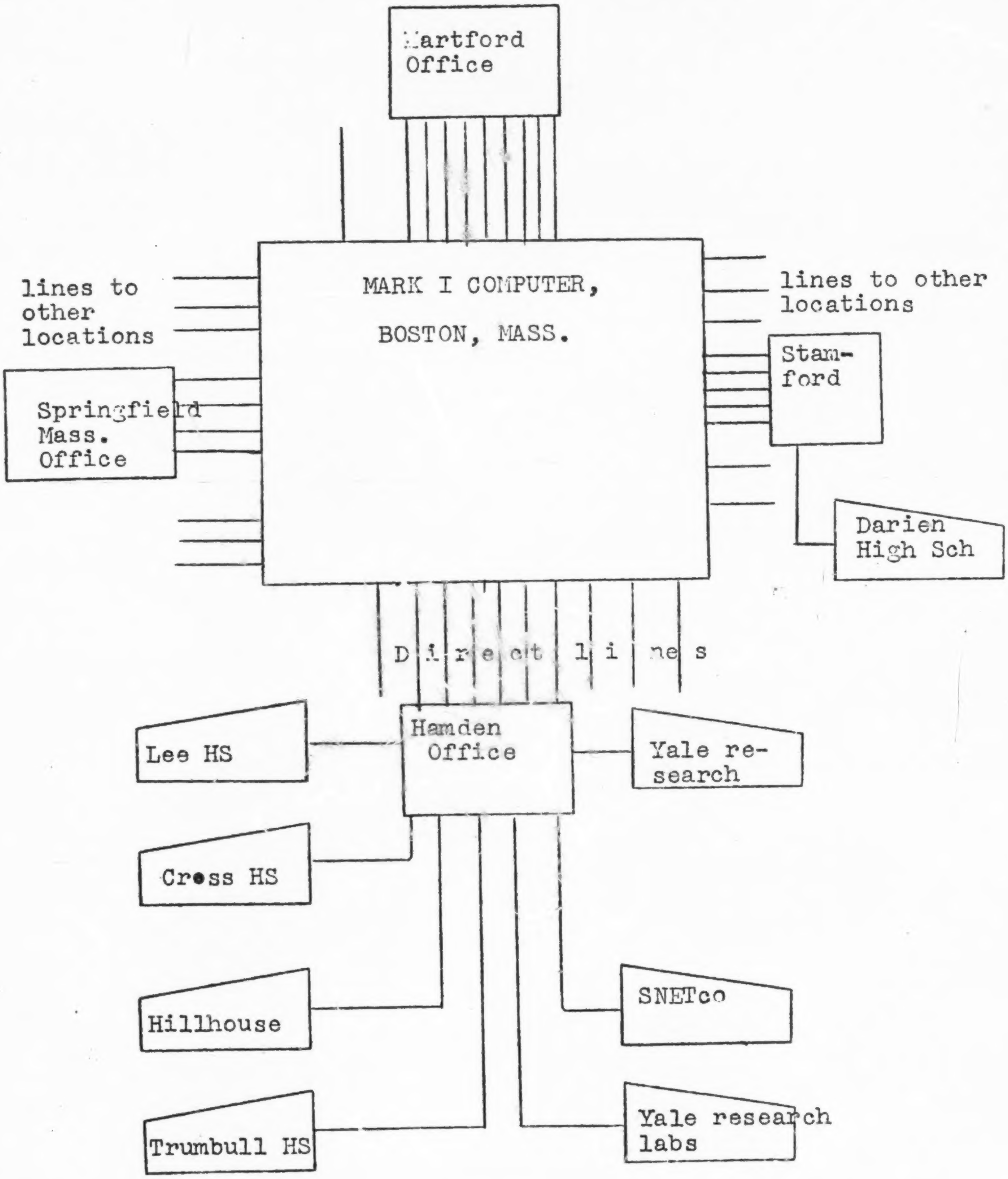
The secret of how we bring this huge and expensive computer in your classroom is the most modern kind of computer technology. It is called time-sharing. It is only in the last few years that it has become possible to do this.

Let's see how time sharing works. First of all we start with the machine that you see right here in the classroom. It is called a TELE-TYPE. If you can look at this word, the way I've broken it up, you will recognize two words. The first is - telephone! The second is - typewriter! That's just what a teletype machine is. It is an electric typewriter that uses telephone lines to connect to any other teletypewriter in the world. Perhaps some of you may have seen the teletypewriters in hotel offices, the bus terminal, at Western Union, or at the newspaper. If you are connected to one in California, it will type out just what you are typing. Not only can teletypewriters be used to operate other teletypewriters, but they can be used to operate any computer in the world which is set up to be connected by telephone lines to such a device.

The way we use this teletypewriter is shown in the diagram (see Figure). We dial a local telephone number (GE Hamden office). The local office has a direct wire to the computer in Massachusetts, near Boston (Mark I Computer, Wellesley, Massachusetts. The company has three computers there to take care of all of New England. Connecticut uses a little bit more than one of them).

Now we come to time-sharing and how it works. As you can see from the drawing, the computer has many connections to teletype machines all over Connecticut. There are 40 ports in the computer, so that as many as 40 different persons can use the computer at the same time. (Actually, at most only about 15-20 people use the computer during the rush hour, which is from 10:00 a.m. to 2:00 p.m. on weekdays.)

The reason why time-sharing works is the amazing speed of the computer. It (Mark I) can do an arithmetic problem in about .0003 seconds. In other words, if we give it an arithmetic problem like 2 x 2, or 512 x 1676, the computer can do about 3,000 of these and give the answers in about a second. Perhaps you can see that there is no point in using a computer for a month or a day or even an hour if it takes but a few seconds for it to do our problems. When you break down the cost of the computer by seconds, that $144 per hour now becomes only 4¢ per second. If we could just use the computer for the few seconds that we need it, the cost would come down to something that we can afford. This is where time-sharing comes in. The computer works like a busy waitress at a lunch counter. She walks over to you and takes your order, then takes


Hartford Office
MARK I COMPUTER, BOSTON, MASS.
lines to other locations
lines to other locations
Springfield Mass. Office
Stamford
Darien High Sch
Direct lines
Hamden Office
Lee HS
Yale research
Cross HS
Hillhouse
SNETco
Trumbull HS
Yale research labs

The Mark I computer and a drawing of the connections to a few typical users. Direct lines go to branch offices. Local users can call nearby branches to get on-line.

it back to the cook who prepares it. Then she goes to another person and takes his order, then another . . . and another. This way although she only can take one order at a time, if she is quick she can keep more than a dozen people at the lunch counter satisfied with her service. (Another example of this is the master chess player who plays a match against 20 different people,all at once. Actually he only plays one person at a time. However, he is so quick that he can walk quickly back and forth from one person to the other,decide what move to make, make it, take care of the other 19 players, then come back to make his next move shortly after the person has made his.).

This is just the way the time-sharing computer takes care of all its customers, one right after the other. When we use the computer, it usually seems as if we have it all to ourself. We give it a command and it obeys immediately. Sometimes it says "wait", but we seldom have to wait more than a few seconds.

Is a Computer Smart? Computer Programmers.

Now we come to a question, " is a computer smart?" (Usually someone in the class knows the answer to this question). Yes, that is right. A computer is really very stupid. In fact is is far more stupid than the dumbest person in this school. If you were to tell a person to go to the office, he would know how to do it. With a computer, you would have to tell it how many steps to take this way, then to turn, then to walk so many steps to the door, then to stop, then to turn the door knob 1/4 of a turn to the right (if you didn't tell it to stop at the end of a 1/4 of a turn it would probably twist the door knob off), then how far to pull the door open, then to walk so many steps out into the hall, then to turn either to the right or left, and just how many steps to go to the principal's office. The computer is so stupid that if you just told it to walk it would smash right into the door. Yes, the computer is stupid. It has to be told every tiny step to take. The list of orders that we give the computer is called a program. The person who writes the program is called a computer programmer. The man whose picture you saw on page 18 of the book, "You/and the Computer" is called a computer operator. Many times, computers are run on a different system than time-sharing. The computer operator takes a big stack of punched cards (see the top of page 16 for a picture of a picture of a punched card) and drops them in to the card reader (see page 19, the piece of hardware with a number 3 underneath it). People who make up the punched cards are called keypunch operators. For every computer programmer and every computer operator there are many jobs open for keypunch operators.

Let's get back to our computer program. If you had to spend all day writing computer programs in the language of the stupid computer (this is called assembly or machine language), you would soon be climbing up the walls. Practically no one has the patience to write a program this way. Instead, computer experts have invented a way of translating of instructions from the language we use, English or simple mathematical expressions, into the language of the computer. The translating language which we are going to use today is called BASIC. If you are interested in learning how to write programs in BASIC, there is a computer club in the school which is open to all students. We have found that all students, regardless of how advanced they are in mathematics, can learn to write programs in BASIC. Now let us turn to a demonstration of the computer at work. I shall dial up the telephone number of the computer, put in the user number (which is confidential information). Then one of you will take over and operate the computer yourself. (At this point the demonstration begins). You are now "on-line".

Please notice when you operate the teletypewriter, that the computer is too stupid to know the difference between "oh" and "zero". Perhaps you never noticed it, but a "zero" is a kind of long skinny "oh" while the "oh" is rounder and fatter. On this teletype, type "Ø" for "oh" and 0 for "zero". Those of you who have used a typewriter before will see that the keyboard is otherwise just the same as a standard typewriter. However all the letters are capitals (upper case). The shift key is only used when there are 2 symbols on the same key. When you wish to start a new line, as you always do after you have answered the computer's question, remember to push the "return" button. This will start a new line. The computer automatically advances the roller for you. When you are on "local" (typing off - line) you have to press the line feed button to start a new line. Also note that the number 1 is on the upper row of keys. You do not use a lower case L to type a "1". If you do, it will confuse the computer.

Some of you may have wondered what this paper tape does. (A picture of the paper tape is shown on page 16.) You can see that each row of dots which is punched out of the tape stands for a certain letter or a number.We can use the teletype machine for punching out paper tape. We just turn on the paper tape punch and type the message we wish. The device called the tape reader reads the tape and runs the teletypewriter at a rate of 100 words per minute. Since the cost of running a teletype terminal is expensive ($7.50 per hour or more while "on - line") we save money by putting in our programs on paper tape. Instead of typing at only a few words per minute, our message goes in at 100 words per minute. For today's demonstration, we do not need to feed the program in. It is stored in the computer's memory. It is a "old" program.

APPENDIX

DETAILS OF GENERAL ELECTRIC TIME SHARING S

Name of Computer	Manufacturer's Designation	Price	Approximate time To do an arithmetic problem	 To ask a and rec. (use of mode)
Mark I	G.E. 235-Datanet 30, Model 265 time-sharing system	About 1/2 million dollars rental- $16,000 month, $144 a day, 4 ¢ per sec.	.0003 sec. (300 μ sec.)	2 sec.
Mark II	G.E. 635 time-sharing system	About 1-1/2 million dollars rental- $1,080 per hr. 30¢ per sec.	.00015 sec. (15 μ sec.)	.1 sec.
Teletype Corp.	ASR - 33	About $800 rental - $80 per month ($45 for rental of unit and $35 per month for tel. service.) $7.50/hr. on line to Mark I* $10.00/hr. on line to Mark II*	----------	-------

*50% off after 6 P.M. and on Saturday.

The Assignment Sheets for the Course

These are self-explanatory. You will have completed the course when you do the 7th assignment. All assignments are done on the computer.

Section III - Demonstration Program Library

These sheets, plus items 5 and 6 of Section I can be used as curriculum and user's guides for BASIC programming. Copies of these can be obtained from W. Lichten. A complete set of tapes is on stock at this office. To get copies of the notes, or a tape, call 787-3131, ext. 2366, on days that Mr. Lichten is not at North Haven.

Revised 6/16/70

Index to Program Library

A2Z - (See NUM)

BACTERIA - A Biology Program

BASIC PROGRAMMING - How to Write a Program to Add or Subtract;
How to Print with the Computer

BATNUM

CHECK

CØMMON - A program to find greatest common divisor of three numbers - Subroutines

DATA PROCESSING IN BASIC - Loops

DATA PROCESSING - Branches

DATA PROCESSING IN BASIC - A bank statement - string variables

DENSE - A program to calculate volume and density for physical science

FACTØR - Finds the Prime Factors of an integer

FIL - (See INV)

FUN

FUNCTION PROGRAMS

HITHAR - A simple illustration of the use of subroutines

HOW TO WRITE A COMPUTER PROGRAM IN BASIC (A one page Basic Programming Manual)

INV and FIL - Lists and Tables - Data Files

KENNY - Illustrates arithmetic operations

L.C.M. - Finds the least common multiple and greatest common factor of two integers

MIRRØR - A program for physics - solves the mirror equation

MIXER - Tells how to mix up a solution - for Chemistry and Biology

NIM - A game which involves binary numbers

NUM, ØRD and A2Z - Filing with the Computer

PI- Three forms of input

PRINT - Illustrates the print command and use of subscripted variables

QUADEQ - Solves quadratic equations

SHØES - Doubly subscripted variables, the TAB command

SPEEDY - Finds percentages quickly

TAKHØM - Makes up a paycheck - Data Banks - Matrices

TEMP - A Physical Science program to convert one temperature scale into all three - Fahrenheit, Celsius and Kelvin

TRIGONOMETRIC PROGRAMS - RTTRI and TRIANG***

WASTE - A program to calculate class averages

Revised 6/16/69

Course in BASIC Programming - Assignments

Lesson I - Getting Started - Simple Printing and Arithmetic

Preparation: For general background, read

A - "You/ and the Computer" (General Electric)

B - "Teacher's Guide for Classroom Demonstration of the Computer" (in section I of this handbook)

Laboratory work:

A - Learn to punch tape on local (see Learner's Guide, page 19)

B - Write a few programs to do simple arithmetic operations and to print a short message (See "How to Write a Program In BASIC in the Program Library). Put your program on tape. Then try it out on the computer (See Learner's Guide, page 19)

C - Read the BASIC Programming sheets in the Program Library. Do the problems on page 5 of the BASIC programming sheets.

D - Read KENNY in the Program Library. Write a similar program, put into tape, and run it off.

Lesson II - Input - Getting Data into the Computer

Preparation: Read

A - Rules for the Use of the Computer and How to Write a Program in section I of the Handbook. Note especially the rules for saving programs and The Panic Button (how to get out of a jam).

B - Learner's Guide, especially the sample problems on pages 16-18.

C - PI (in the Program Library)

D - DATA processing in BASIC (Program Library). Bring in a sales slip to the laboratory.

Laboratory:

A - Complete any work from assignment I that you didn't finish.

B - Take the TEST to show that you know how to end a run.

C - Do the assignment at the end of PI.

D - If time permits, do the homework assignment at the end of BILL.

Lesson III - Branches and Loops - Preparing a Bank Statement

Preparation: Read

A - Reference Manual, pp. 2-23. Much of this will be review.

B - Program Library

DATA Processing - Branches
DATA Processing in BASIC - A bank Statement - String Variables.
QUADEQ

Laboratory:

Write a bank statement of your own. Improve the format of the program in the library if you can. (You could have separate columns for deposits and checks, for example.)

Lesson IV - Lists and Tables - Data Files - Filing with the Computer

Preparation: Go as far as you can in the following list of programs:

A - Inventory Control - Lists and Tables - Data Files (in the program library)

References: Reference Manual, pp. 9-11, 15-16, 18.

B - Sorting and Ordering Lists - NUM, ØRDER, and A2Z - Filing with the Computer (program library).

Reference: Reference Manual, pp. 37-40.

C - Lists with Columns and Tables (Matrices)

SHOES

PRINT

TAKHØM, all in the reference library

References: Reference Manual, pp. 21-23,
Material on Matrices and Dimensioning, pp. 48-49, bottom of p. 51.

D - As a further example of lists, see MIXER.

Lesson V - Subroutines, Functions, Defined Functions

Preparation: Read

A - HITHAR and
CØMMØN in the program library

References: Reference Manual, pp. 32-33 (GØSUB)

If time permits

B - For functions and defined functions, read pp. 5 - 6, and p. 21 in the Reference Manual.

TRIANG*** (See Trigonometric Programs in the Library) has an example of a defined function.

Laboratory work:

Write a laboratory program that uses subroutines. Here are a few suggestions:

1 - Take your program to write a bank statement. Put in some GØSUB statements to put a row of asterisks and put a triple space before and after each statement. Also, if you have a separate statement of balance carried forward, or final balance, set it off neatly.

2 - Write a program to find the lowest common multiple of three numbers, using L.C.M. as a subroutine.

3 - Write a program to find the G.C.F. of any number of integers. Use CØMMØN as a guide. Use WASTE as a guide for handling the DATA statement.

4 - Write a program to make change automatically. For Example, the change for a $.23 item from a dollar is 50¢, 25¢, 2 x 1¢. (See page 201 of "A FORTRAN IV Primer," by Organick.)

BACTERIA, A Biology Program

Population growth and genetics, with their mathematical relationships, are natural areas for computer usage in biology. BACTERIA is a simple program which calculates the growth rate of a colony of bacteria. Of course, the remarkable result here is the immense numbers that can occur in a short time.

A sample run is shown for a class in biology at Yale-North Haven High School. Note that the student first got an immense number that was too large, even for the computer to handle, when she chose a division rate of one each minute and a 24 hour period. A more modest time requirement of one hour still gave an immense number.

The program is written to handle two conditions. If the number of bacteria contains more than eight digits, the computer prints out the results in scientific notation. In this case (in line 270 of the list), the program provides for a short explanation of scientific notation. In case the number of bacteria exceeds 2^{127}, there would be an overflow condition in the computer. The instruction in line 205 avoids this. The error message "There will be too many cells for even me to count" is typed out.

```
BACTERIA       10:45       07/25/69

BACTERIA REPR0DUCE BY SPLITTING IN TW0.  I'LL TELL Y0U H0W
QUICKLY A C0L0NY 0F BACTERIA GR0WS.  IT STARTS WITH 0NLY 0NE
CELL.  Y0U TELL ME H0W 0FTEN THE CELL DIVIDES AND I'LL TELL
Y0U H0W MANY CELLS THERE WILL BE AT ANY TIME.

H0W MANY MINUTES D0ES IT TAKE A CELL T0 DIVIDE? 1
H0W MANY H0URS D0 Y0U WANT T0 LET IT MULTIPLY? 24

IF A CELL DIVIDES EVERY  1 MINUTE(S), AFTER  24 H0UR(S)
THERE WILL BE T00 MANY CELLS F0R EVEN ME T0 C0UNT.
```

```
DØ YØU WANT ANØTHER LENGTH ØF GRØWTH TIME (YES ØR NØ)? YES
HØW MANY HØURS DØ YØU WANT TØ LET IT MULTIPLY? 1

IF A CELL DIVIDES EVERY  1 MINUTE(S), AFTER  1 HØUR(S)
THERE WILL BE  1.15292E+18 CELLS.

FØR LARGE NUMBERS, I PRINT MY RESULTS IN 'SCIENTIFIC NØTA-
TIØN'.  THUS, 2E+12 MEANS 2*10↑12, ØR 2,000,000,000,000.

DØ YØU WANT ANØTHER LENGTH ØF GRØWTH TIME (YES ØR NØ)? NØ
DØ YØU WANT A DIFFERENT DIVISIØN TIME (YES ØR NØ)? NØ

MY RESULTS WERE CØRRECT PRØVIDING THAT THERE WAS PLENTY ØF
FØØD AND SPACE FØR THE NEW BACTERIA, AND NØ ENEMIES.  IN
REAL LIFE, THESE FACTØRS WØULD KEEP THE PØPULATIØN DØWN.

I HAVE ENJØYED HELPING YØU.  CØME AGAIN.  GØØDBYE.


USED      .32 UNITS
REPLACE
READY
RUN

BACTERIA      10:48        07/25/69

BACTERIA REPRØDUCE BY SPLITTING IN TWØ.  I'LL TELL YØU HØW
QUICKLY A CØLØNY ØF BACTERIA GRØWS.  IT STARTS WITH ØNLY ØNE
CELL.  YØU TELL ME HØW ØFTEN THE CELL DIVIDES AND I'LL TELL
YØU HØW MANY CELLS THERE WILL BE AT ANY TIME.

HØW MANY MINUTES DØES IT TAKE A CELL TØ DIVIDE? 5
HØW MANY HØURS DØ YØU WANT TØ LET IT MULTIPLY? 2

IF A CELL DIVIDES EVERY  5 MINUTE(S), AFTER  2 HØUR(S)
THERE WILL BE  16777216 CELLS.

DØ YØU WANT ANØTHER LENGTH ØF GRØWTH TIME (YES ØR NØ)? YES
HØW MANY HØURS DØ YØU WANT TØ LET IT MULTIPLY? 8

IF A CELL DIVIDES EVERY  5 MINUTE(S), AFTER  8 HØUR(S)
THERE WILL BE  7.92282E+28 CELLS.

FØR LARGE NUMBERS, I PRINT MY RESULTS IN 'SCIENTIFIC NØTA-
TIØN'.  THUS, 2E+12 MEANS 2*10↑12, ØR 2,000,000,000,000.

DØ YØU WANT ANØTHER LENGTH ØF GRØWTH TIME (YES ØR NØ)? NØ
DØ YØU WANT A DIFFERENT DIVISIØN TIME (YES ØR NØ)? NØ

MY RESULTS WERE CØRRECT PRØVIDING THAT THERE WAS PLENTY ØF
FØØD AND SPACE FØR THE NEW BACTERIA, AND NØ ENEMIES.  IN
REAL LIFE, THESE FACTØRS WØULD KEEP THE PØPULATIØN DØWN.

I HAVE ENJØYED HELPING YØU.  CØME AGAIN.  GØØDBYE.


USED      .22 UNITS
LISTNH
```

```
100 PRINT"BACTERIA REPRØDUCE BY SPLITTING IN TWØ.  I'LL TELL YØU HØW"
110 PRINT"QUICKLY A CØLØNY ØF BACTERIA GRØWS.  IT STARTS WITH ØNLY ØNE"
120 PRINT"CELL.  YØU TELL ME HØW ØFTEN THE CELL DIVIDES AND I'LL TELL"
130 PRINT"YØU HØW MANY CELLS THERE WILL BE AT ANY TIME."
140 PRINT
150 PRINT"HØW MANY MINUTES DØES IT TAKE A CELL TØ DIVIDE";
160 INPUT M
170 PRINT"HØW MANY HØURS DØ YØU WANT TØ LET IT MULTIPLY";
180 INPUT H
190 PRINT
200 PRINT"IF A CELL DIVIDES EVERY ";M;"MINUTE(S), AFTER ";H;"HØUR(S)"
205 IF 60*H/M>127 THEN 230
210 LET N=2↑(INT(60*H/M))
220 GØTØ 250
230 PRINT"THERE WILL BE TØØ MANY CELLS FØR EVEN ME TØ CØUNT."
240 GØTØ 300
250 PRINT"THERE WILL BE ";N;"CELLS."
260 PRINT
270 IF N<10↑8 THEN 310
280 PRINT"FØR LARGE NUMBERS, I PRINT MY RESULTS IN 'SCIENTIFIC NØTA-"
290 PRINT"TIØN'.  THUS, 2E+12 MEANS 2*10↑12, ØR 2,000,000,000,000."
300 PRINT
310 PRINT"DØ YØU WANT ANØTHER LENGTH ØF GRØWTH TIME (YES ØR NØ)";
320 INPUT A$
330 IF A$="YES" THEN 170
340 IF A$="NØ" THEN 370
350 PRINT"PLEASE ANSWER EITHER YES ØR NØ."
360 GØTØ 310
370 PRINT"DØ YØU WANT A DIFFERENT DIVISIØN TIME (YES ØR NØ)";
380 INPUT B$
390 IF B$="YES" THEN 150
400 IF B$="NØ" THEN 430
410 PRINT"PLEASE ANSWER EITHER YES ØR NØ."
420 GØTØ 370
430 PRINT
440 PRINT"MY RESULTS WERE CØRRECT PRØVIDING THAT THERE WAS PLENTY ØF"
450 PRINT"FØØD AND SPACE FØR THE NEW BACTERIA, AND NØ ENEMIES.  IN"
460 PRINT"REAL LIFE, THESE FACTØRS WØULD KEEP THE PØPULATIØN DØWN."
470 PRINT
480 PRINT"I HAVE ENJØYED HELPING YØU.  CØME AGAIN.  GØØDBYE."
490 END
```

BASIC PROGRAMMING

How to write a program to add or subtract

Suppose you want to solve this problem:
Find the new balance in a checking account with the following:

Previous Balance	Deposits	Checks Issued	New Balance
$700.00	$50.25,$100.00	$25.00,$10.00	?

The solution is found by adding all the deposits and subtracting all the checks from the previous balance:

700.00	old balance
+ 50.25	add deposit
+100.00	add deposit
850.25	balance before checks are subtracted
- 25.00	subtract check
825.25	balance
- 10.00	subtract other check
815.25	new balance

To solve the problem on the computer, after it is "on line" (connected to our teletype machine), we do the following:

What we see

```
NEW
NEW FILE NAME--BANK

READY

10 PRINT 700.00+50.25+100.00-25.00-10.00
20 END


RUN


BANK          13:51  M  MØN  10/21/68

    815.25


RAN            1.33 SEC
```

What we do

We type "NEW"
We type "BANK" when the computer types "NEW FILE NAME--"
When it says "READY", we type in our program. We put a "+" in front of everything we wish to add, a "-" in front of what we wish to subtract. The lines must be numbered in order. The "END" line is last.

We type "RUN", and the computer finds the answer. The rest of what you see has been typed by the computer. The next line gives the time, date and year.
The answer is $815.25

It took 1.33 seconds to do it!

The program is the list of orders we give to the computer. It is inside the box. The program can be used over and over again by the computer.

How to write a program to multiply-

Suppose you want to solve the problem:

At the store, hot dogs are $.59 for one pound. How much do three pounds of hot dogs cost?

We solve the problem by multiplying the number of pounds by the price for each pound:

.59	Cost for each pound of hot dogs
x 3	xnumber of pounds of hot dogs.
1.77	Total cost.

To solve the problem on the computer, after it is "on line" we do the same as before. We write a short program, which is inside the box.

What we see	What we do
NEW NEW FILE NAME--HØTDØG READY 10 PRINT 3*.59 20 END RUN	We type in "NEW" We type a name "HOTDOG" (6 spaces or less). The computer tells us it's ready. We type our program. We put a * between numbers we want to multiply. The lines are numbered in order. The END line is last. Now we type "RUN"
HØTDØG 13:52 M MON 10/21/68	The computer does the rest It types out the time and date.
1.77	The answer is $1.77
RAN 1.33 SEC	It took only 1.33 seconds!

How to write a program to divide

Suppose you want to solve the problem:

At the store, one pound of butter costs $1.08. This is too much for Mrs. Brown. She decides to buy only one quarter of a pound. How much does this cost her?

The solution is found by dividing $1.08 by 4.

1.08	Cost for one pound
÷ 4	divide by four
.27	cost for a quarter pound

To solve the problem on the computer, after it is "on line", we do the same as before. We write a short program, which is inside the box.

What we see

```
NEW
NEW FILE NAME--BUTTER
READY.

10 PRINT 1.08/4
20 END
RUN
WAIT.


BUTTER  15:23   M M0N  10/21/68

 .27



RAN   1.17 SEC
```

What we do

We type in "NEW". We type in a name "BUTTER". The computer tells us it's ready. We type our program. We put a / before the number we are using to divide (the divisor). The lines are numbered in order. The END line is last. Now we type "RUN". The computer says "WAIT" when it is busy.

It types the program name, time and date.

The answer is $.27.

It took only 1.17 seconds!

BASIC PROGRAMMING

How to Print with the Computer

The computer will print whatever you want at any place. For example, in the BANK program, (see page 1) suppose you want to print

```
YOUR NEW BALANCE IS
815.25
```

You can add a line to your program to do this. Just tell the computer to print what you want. The correct program is

```
5 PRINT "YØUR NEW BALANCE IS"
10 PRINT 700.00 + 50.25 + 100.00 -25.00 -10.00
20 END
```

Line Numbering and What It Does

If you decided to print the message after you had your program written, you could have added line 5 like this:

```
10 PRINT 700.00 +50.25 +100.00 -25.00 -10.00
20 END
5 PRINT "YØUR NEW BALANCE IS"
```

The computer *automatically* would rearrange your commands in the order of the line numbers (line 5, then 10, then 20 - just as if you wrote them in that order). If you wrote the program:

```
10 PRINT 700.00 +50.25 +100.00 -25.00 -10.00
15 PRINT "THIS IS YØUR NEW BALANCE."
20 END
```

The computer would print

```
815.25
THIS IS YØUR NEW BALANCE.
```

Note that the message is printed *after* the number, because the command to print the message comes in line 15, *after* the command to print the answer.

We see that it doesn't matter what order we type in the commands on the tape or keyboard. The computer takes the commands in order of the line numbers.

More about Printing

Suppose you want to print $815.25 instead of 815.25. How do you do it? Just retype line 10 like this:

```
10 PRINT "$"; 700.00 +50.25 +100.00 -25.00 -10.00
```

The computer will print $ 815.25.

If you want to print YØUR NEW BALANCE IS $ 815.25, your program should be

```
10 PRINT "YØUR NEW BALANCE IS   $";700.00+50.25+100.00-25.00-10.00;"."
20 END
```

Notice that the semicolon (;) leaves a space between the message between the quotation marks and the next number or message.

If you use a comma instead of a semicolon, the computer leaves a larger space. The program

```
10 PRINT"YØUR NEW BALANCE IS", "$",700.00+50.25+100.00-25.00-10.00;"."
20 END
```

prints

```
YØUR NEW BALANCE IS          $ 815.25  .
```

The Print Command

Notice that whenever you type PRINT, the computer moves to the next line. For example, if you give the command

```
5 PRINT "YØUR NEW BALANCE IS"
10 PRINT
15 PRINT "$"; 700.00+50.25+100.00-25.00-10.00;"."
20 END
```

You'll get

```
YØUR NEW BALANCE IS

$ 815.25 .
```

Notice that the command PRINT makes the computer skip a line. This is how you double space print-out.

Problems for Printing with BASIC

Jimmy Johnson wrote a program to tell us a few vital facts about himself. The print out is shown right below this line. The list of his program is then given. Note that he used all four arithmetic operations.

```
JIMMY     16:10   M  FRI  12/20/68

MY NAME IS JAMES JØHNSØN.
I LIVE AT 121 WASHINGTØN AVE.
I WEIGH 143 PØUNDS.
MY HEIGHT IS 70.5 INCHES.
I AM IN GRADE 10   .

USED      4.33 UNITS.
LISTNH

10 PRINT "MY NAME IS JAMES JØHNSØN."
20 PRINT "I LIVE AT"; 11+10+100; "WASHINGTØN AVE."
30 PRINT "I WEIGH";13*11;"PØUNDS."
40 PRINT "MY HEIGHT IS";141/2; "INCHES."
50 PRINT "I AM IN GRADE"; 27-17; "."
60 END
```

1. Do the same for yourself. Write a program to tell us about your name, address, weight, height, grade, or any other vital statistics.
2. Write a similar program to give someone's address, zip code included.

Revised 12-20-68

BATNUM

"Battle of Numbers" is the old game in which you are given a pile of sticks. Two persons remove sticks until all are taken. The number of sticks taken must equal or larger to a minimum value (always greater than zero; usually equal to one) and must not exceed a maximum value. The theory of this game is given in Kemeny and Kurtz, BASIC Programming.

BATNUM is a program obtained from Dartmouth College to play "Battle of Numbers." It is written in a variant of BASIC which can be used on the G E Mark II computer or the Dartmouth computer. Lines 100-200 are REM statements (see the Mark II Reference Manual, p. 24). If these lines are deleted, the program can be used on any computer which handles BASIC. The directions printed out in each run are self explanatory.

A copy of the list and a sample run are shown.

```
100'  NAME--BATNUM
110'
120'  DESCRIPTIØN--BATTLE ØF NUMBERS BETWEEN USER AND CØMPUTER
130'
140'  SØURCE--UNKNØWN
150'
160'  INSTRUCTIØNS--TYPE "RUN" AND FØLLØW INSTRUCTIØNS.
170'
180'
190'  * * * * * *  MAIN PRØGRAM * * * * * * * * * * *
200'
210 PRINT"THIS DEMØNSTRATIØN SHØWS YØU SØME VERSATILE FEATURES ØF"
220 PRINT"GE'S TIME-SHARING SERVICE. IT IS A 'BATTLE-ØF-NUMBERS'"
230 PRINT"GAME, WHERE THE CØMPUTER IS YØUR ØPPØNENT."
240 PRINT
250 PRINT"THE GAME STARTS WITH AN ASSUMED PILE ØF ØBJECTS.YØU AND"
260 PRINT"YØUR ØPPØNENT ALTERNATLY REMØVE ØBJECTS FRØM THE PILE."
270 PRINT"WINNING IS DEFINED IN ADVANCE AS TAKING THE LAST ØBJECT"
280 PRINT"ØR NØT. YØU CAN ALSØ SPECIFY SØME ØTHER BEGINNING"
290 PRINT"CØNDITIØNS. DØN'T USE ZERØ, HØWEVER, IN PLAYING THE GAME."
300 PRINT
310 PRINT
320 GØTØ 440
330 PRINT
340 PRINT
350 PRINT
360 PRINT
370 PRINT
380 PRINT
390 PRINT
400 PRINT
410 PRINT
420 PRINT
430 PRINT
440 PRINT"ENTER PILE SIZE:";
450 INPUT N
460 IF N<>0 THEN 480
470 GØTØ 440
480 IF N<>INT(N) THEN 330
490 IF N<1 THEN 330
500 PRINT "ENTER WIN ØPTIØN - 1 TØ TAKE LAST, 2 TØ AVØID LAST: ";
510 INPUT M
520 IF M=1 THEN 540
530 IF M<>2 THEN 500
540 PRINT "ENTER MIN AND MAX: ";
550 INPUT A,B
560 IF A>B THEN 540
570 IF A<1 THEN 540
580 IF A<>INT(A) THEN 540
590 IF B<>INT(B) THEN 540
600 PRINT"ENTER START ØPTIØN - 1 CØMPUTER FIRST, 2 YØU FIRST:";
610 INPUT S
620 IF S=1 THEN 640
630 IF S<>2 THEN 600
640 LET C=A+B
650 IF S=2 THEN 680
660 GØSUB 710
670 IF W=1 THEN 330
680 GØSUB 980
```

```
700 GO TO 660
710 LET Q=N
720 IF M=1 THEN 740
730 LET Q=Q-1
740 IF M=1 THEN 790
750 IF N>A THEN 830
760 LET W=1
770 PRINT"COMPUTER TAKES";N;"AND LOSES."
780 RETURN
790 IF N>B THEN 830
800 LET W=1
810 PRINT"COMPUTER TAKES";N;"AND WINS."
820 RETURN
830 LET P=Q-C*INT(Q/C)
840 IF P>=A THEN 860
850 LET P=A
860 IF P<=B THEN 880
870 LET P=B
880 LET N=N-P
890 PRINT"COMPUTER TAKES";P;"AND LEAVES";N
900 LET W=0
910 RETURN
920 PRINT "YOUR MOVE: ";
930 INPUT P
940 IF P<>0 THEN 980
950 PRINT"I TOLD YOU NOT TO USE ZERO...COMPUTER WINS BY FORFEIT."
960 LET W=1
970 RETURN
980 IF P<>INT(P) THEN 1030
990 IF P>=A THEN 1020
1000 IF P=N THEN 1070
1010 GO TO 1030
1020 IF P<=B THEN 1050
1030 PRINT "ILLEGAL MOVE, REENTER IT: ";
1040 GO TO 930
1050 LET N=N-P
1060 IF N<>0 THEN 1140
1070 IF M=1 THEN 1110
1080 PRINT"TOUGH LUCK, YOU LOSE."
1090 LET W=1
1100 RETURN
1110 PRINT"CONGRATULATIONS, YOU WIN."
1120 LET W=1
1130 RETURN
1140 IF N>=0 THEN 1170
1150 LET N=N+P
1160 GO TO 1030
```

```
1170 LET W=0
1180 RETURN
1190 END
RUN
WHAT?
RUN

BATNUM        16:39        06/25/69

THIS DEMØNSTRATIØN SHØWS YOU SØME VERSATILE FEATURES ØF
GE'S TIME-SHARING SERVICE. IT IS A 'BATTLE-ØF-NUMBERS'
GAME, WHERE THE CØMPUTER IS YØUR ØPPØNENT.

THE GAME STARTS WITH AN ASSUMED PILE ØF ØBJECTS.YØU AND
YØUR ØPPØNENT ALTERNATLY REMØVE ØBJECTS FRØM THE PILE.
WINNING IS DEFINED IN ADVANCE AS TAKING THE LAST ØBJECT
ØR NØT. YOU CAN ALSØ SPECIFY SØME ØTHER BEGINNING
CØNDITIØNS. DØN'T USE ZERØ, HØWEVER, IN PLAYING THE GAME.


ENTER PILE SIZE:? 5
ENTER WIN ØPTIØN - 1 TØ TAKE LAST, 2 TØ AVØID LAST: ? 2
ENTER MIN AND MAX: ? 1,2
ENTER START ØPTIØN - 1 CØMPUTER FIRST, 2 YØU FIRST:? 2
YØUR MØVE: ? 1
CØMPUTER TAKES 1 AND LEAVES 3
YØUR MØVE: ? 2
CØMPUTER TAKES 1 AND LØSES.


ENTER PILE SIZE:? STØP
PRØGRAM HALTED

USED     .72 UNITS
```

This program was written by David Howell, Lee High School.

```
CHECK     12:12     M    FRI 10/11/68


CHECKING ACCØUNT STATEMENT

ENTER ØLD BALANCE? 382.16

ARE THERE ANY DEPØSITS?  YES

HØW MANY DEPØSITS ARE THERE?  3
ENTER ØNE ØF YØUR   3   DEPØSITS AFTER EACH'?'.
? 16.83
? 22.95
? 11.07

ARE THERE ANY CHECKS? YES

HØW MANY CHECKS ARE THERE? 1
ENTER ØNE ØF YØUR   1   CHECKS AFTER EACH '?'
? 223.67
YØUR NEW BALANCE IS $ 209.34

DØ YØU WANT TØ DØ ANØTHER PRØBLEM? NØ

I AM GLAD I CØULD HELP YØU IN YØUR CØNSUMER MATHEMATICS
CLASS.  BUT I CØULD NØT HAVE HELPED AT ALL IF YØU HAD
NØT FIGURED ØUT HØW TØ DØ THE PRØBLEM.

SEE IF YØU CAN DESIGN A CØMPUTER TØ DØ ØTHER PRØBLEMS.

SEE YA ARØUND.  KEEP UP THE GØØD WØRK.
RAN    14.00  SEC
LISTNH

 10 LET T = 0
 20 LET C = 0
090 PRINT
100 PRINT "CHECKING ACCØUNT STATEMENT"
110 PRINT
120 PRINT "ENTER ØLD BALANCE";
125 INPUT B
130 PRINT
135 PRINT "ARE THERE ANY DEPØSITS";
140 INPUT A$
150 IF A$ = "YES" THEN 200
155 IF A$ = "NØ" THEN 300
160 GØTØ 800
200 PRINT
205 PRINT "HØW MANY DEPØSITS ARE THERE";
210 INPUT N
215 PRINT "ENTER ØNE ØF YØUR ";N;" DEPØSITS AFTER EACH '?'."
```

```
220 FOR I = 1 TØ N
225 INPUT D(I)
230 LET T = T + D(I)
235 NEXT I
300 PRINT
305 PRINT "ARE THERE ANY CHECKS";
310 INPUT A$
315 IF A$ = "YES" THEN 400
320 IF A$ = "NØ" THEN 500
325 GØTØ 810
400 PRINT
405 PRINT "HØW MANY CHECKS ARE THERE";
410 INPUT N
415 PRINT "ENTER ØNE OF YØUR ";N;" CHECKS AFTER EACH '?'."
420 FØR I = 1 TO N
425 INPUT P(I)
430 LET C = C + P(I)
435 NEXT I
500 PRINT "YØUR NEW BALANCE IS $ "; B + T - C
501 PRINT
600 PRINT "DØ YØU WANT TØ DØ ANØTHER PRØBLEM";
610 INPUT A$
620 IF A$ = "YES" THEN 10
625 IF A$ = "NØ" THEN 990
650 PRINT "I DIDN'T UNDERSTAND YØU. CØULD YØU PLEASE ANSWER AGAIN?"
660 PRINT "ANSWER 'YES' OR 'NØ' THIS TIME, PLEASE."
670 GØTØ 501
800 PRINT "PLEASE TYPE MØRE CAREFULLY. ANSWER 'YES' OR 'NØ'."
802 GØTØ 130
810 PRINT "YØU MUST ANSWER THE QUESTIØN WITH 'YES' OR 'NØ'."
811 PRINT "PLEASE TYPE CAREFULLY."
812 GØTØ 300
990 PRINT
991 PRINT "I AM GLAD I CØULD HELP YØU IN YØUR CØNSUMER MATHEMATICS"
992 PRINT "CLASS. BUT I CØULD NØT HAVE HELPED AT ALL IF YOU HAD"
993 PRINT "NØT FIGURED ØUT HØW TØ DØ THE PRØBLEM."
994 PRINT
995 PRINT "SEE IF YØU CAN DESIGN A CØMPUTER TØ DØ ØTHER PRØBLEMS."
996 PRINT
997 PRINT "SEE YA ARØUND. KEEP UP THE GØØD WØRK."
999 END
```

CØMMØN, A Program to find the Greatest Common Divisor of Three Numbers. Subroutines

Often one has to do the same operation at several points in a computer program. Rather than lengthen the program by writing the same commands over again, one uses a subroutine. A subroutine is a part of a computer program that can be used more than once.

CØMMØN uses the Euclidean Algorithm to find the greatest common divisor, G, of two numbers, A and B. It then finds the greatest common divisor of G and a third number C. This number is the greatest common divisor of all three numbers: A, B and C.

The subroutine, in lines 200-300, is the Euclidean Algorithm. On line 50, the program says "GØSUB 200", which has the same effect as "GØTØ 200". After the subroutine runs, the RETURN statement in line 310 tells the computer to go back to line 60, which follows the GØSUB statement in line 50. Line 80 is another GØSUB statement, which sends the computer back to the subroutine 200-300. The RETURN statement now tells the computer to go back to line 90, which finishes the job of finding the greatest common factor.

Lines 100-150 give the user a chance to do another problem.

The subroutine can go at any place in the program. Note, however, the requirement a GØTØ statement at the right place (line 130) to make sure that the subroutine is reached only by a GØSUB statement. If the user wants to stop, line 130 tells the computer to go safely past lines 200-310 to the graceful exit of lines 320-999.

A list of the program and sample run are shown.

```
1 REM THIS PRØGRAM ILLUSTRATES THE USE ØF SUBRØUTINES.  THE SUBRØUTINE
2 REM IS IN LINES 200-300.
5 PRINT"THIS IS A PRØGRAM TØ CØMPUTE THE GREATEST CØMMØN FACTØR ØF"
6 PRINT "THREE INTEGERS, A,B,C. TYPE IN YØUR VALUES FØR A,B,C. SEPARATE"
7 PRINT"THE NUMBERS FRØM EACH ØTHER BY A CØMMA, LIKE THIS:2,6,12 'RETURN'"
8 PRINT
10 PRINT"WHAT ARE YØUR VALUES FØR A,B,C";
20 INPUT A,B,C
25 PRINT
26 PRINT "A","B","C","G.C.F."
27 PRINT
30 LET X = A
40 LET Y = B
50 GØSUB 200
60 LET X = G
70 LET Y = C
80 GØSUB 200
90 PRINT A,B,C,G
95 PRINT
100 PRINT "DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)";
110 INPUT A$
120 IF A$ = "YES" THEN 8
130 IF A$ = "NØ" THEN 320
140 PRINT "SØRRY, I DIDN'T GET YØUR REMARK. PLEASE RETYPE IT."
150 GØTØ 100
200 LET Q = INT(X/Y)
210 LET R = X-Q*Y
220 IF R = 0 THEN 300
230 LET X = Y
240 LET Y = R
250 GØTØ 200
300 LET G = Y
310 RETURN
320 PRINT "GØØDBYE. CØME BACK AGAIN. NØW SAY GØØDBYE TØ ME."
999 END

RUN

CØMMØN    15:22    M    TUE  12/10/68

THIS IS A PRØGRAM TØ CØMPUTE THE GREATEST CØMMØN FACTØR ØF
THREE INTEGERS, A,B,C. TYPE IN YØUR VALUES FØR A,B,C. SEPARATE
THE NUMBERS FRØM EACH ØTHER BY A CØMMA, LIKE THIS:2,6,12 'RETURN'

WHAT ARE YØUR VALUES FØR A,B,C? 12,32,96
```

```
A                   B                   C                   G.C.F

 12                  32                  96                  4

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? YES

WHAT ARE YØUR VALUES FØR A,B,C? 24,108,1728

A                   B                   C                   G.C.F.

 24                  108                 1728                12

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? YES

WHAT ARE YØUR VALUES FØR A,B,C? 3219,8103,10767

A                   B                   C                   G.C.F.

 3219                8103                10767               111

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? NØ
GØØDBYE. CØME BACK AGAIN. NØW SAY GØØDBYE TØ ME.

USED      15.67 UNITS.
```

DATA PROCESSING IN BASIC

In this lesson, we shall learn how to write a program to "process data." What is "data processing"?

Suppose we work at a bank and have to make up monthly statements for each of a thousand people who have checking accounts. This would run into thousands of checks each day. In a year we might have a million checks! To avoid all the labor of adding and subtracting deposits and checks, we write a single program which handles all accounts. The numbers which go with checks and deposits are called data. Data processing is what we do when we handle all these numbers and figure out the new balance of the checking account.

Let's see how to write such a program. For our DATA, we take the bill for food at the supermarket. We'll name our program BILL. The DATA can go in any line in the program. Usually we put these lines near the end of the program. Let's save lines 500-998 for DATA, and call 999 the END statement. We enter our data as numbers separated by commas.

For example, if we buy

one quart of milk	.30
one pound of butter	.90
one loaf of bread	.25
four pounds of steak (at $.90/pound)	3.60
Our total bill is	5.05

Let's do it on the computer. The DATA are on line 500 (If there were more DATA than could fit on one line, we just use more lines of DATA):

```
500 DATA .30, .90, .25, 3.60
999 END
```

Now we have to write the rest of our program. First we tell the computer to read the data. To do this, we have to give a name to the numbers in the DATA. This name can be any letter, such as X. (Also it could be a letter followed by a number from one to ten, such as A1, B3, X5, etc.)

Our READ command is

```
10 READ X
```

Next we tell the computer to add X on to the bill, which we call B.

```
20 LET B = B+X
```

This command may seem strange to you. In the computer, LET B = B+X means take the old value for B,add X to it, and let this equal the new value of B. This is just what we do when we add up a bill.

Let's tell the computer to print X and B:

```
30 PRINT X, B
```

Now, since we have to tell the computer every little step, we must tell it not to stop after it has read the item and added it on the bill. We must tell it to start all over again and do the same for the next item:

```
40 GØTØ 10
```

The computer goes back, reads the next item, adds it on to the bill and keeps going until it finds no more DATA. Whenever this happens, the computer stops and prints "OUT OF DATA IN LINE 10" (line 10 is the line that told the computer to read the data.)

We can improve our program by adding a neat title with a space:

```
5 PRINT "CHARGES", "TØTAL BILL"
6 PRINT
```

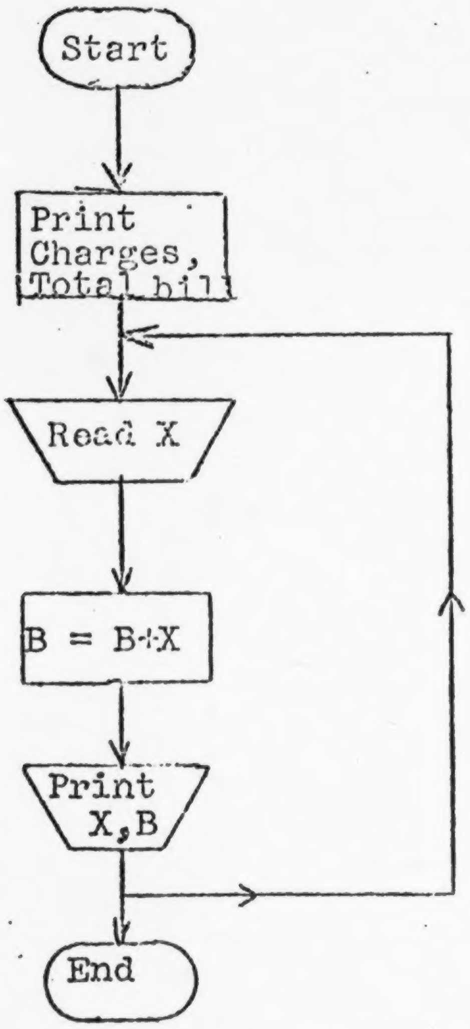
A sample run and a list of the program are shown below. This is our first example of data processing.

```
BILL      11:56    M TUE 12/03/68

CHARGES        TØTAL BILL

 .3             .3
 .9             1.2
 .25            1.45
 3.6            5.05

ØUT OF DATA IN 10



USED      1.83 UNITS.
LIST


BILL      11:56    M TUE 12/03/68

5 PRINT "CHARGES", "TØTAL BILL"
6 PRINT
10 READ X
20 LET B = B+X
30 PRINT X, B
40 GØTØ 10
500 DATA .30, .90, .25, 3.60
999 END
```

Do it yourself! Get a sales slip from a store, write a program like BILL, and check the answer.

FLOW CHARTS

A flow chart is a diagram which shows how the program is written. It helps to draw a flow chart before you write your program. A flow chart for BILL is shown in the figure. Note how we go back to the READ statement. Such a round trip is called a *loop*. Loops are a very important part of any program for data processing, and in many other types of programs.


Start
Print Charges, Total bill
Read X
B = B+X
Print X,B
End

BILL, AN EXAMPLE OF DATA PROCESSING ON THE COMPUTER

DATA PROCESSING - BRANCHES

In the last lesson on data processing, we learned how to put a loop in our program, to let us go back and do the same thing over and over again until the computer is out of data.

In this lesson, we learn about branches in a program. Take our example of BILL. If people only bought things and never paid for them, our programming problems would be simple, but the company would go out of business. People do have to pay their bills. We have to write a program to let them pay up!

We write a program to make up a bill. The program should do the following:

1. read and print the unpaid balance carried forward from the last bill
2. print each charge, add it to the bill and print the new total
3. allow for payments by subtracting them, and finally
4. if the customer overpays his bill, to show him that he has a credit.

Let's first solve our problem by hand. Suppose we owe the department store $10.50 at the beginning of the month. We buy a dress shirt for $4.30, then a blouse for $5.60. Then we send in a payment of $10.50 for last month's bill and return a $10.00 pair of shoes for credit. Then we buy a boy's suit for $24.40. What is our bill?

Our Balance Forward is	10.50
We add on the charges for the shirt.	4.30
and for the blouse:	5.60
balance	20.40

Now we subtract our payment: -10.50

balance 9.90

and try to subtract our refund for the shoes: -10.00

?

But it won't work! What do we do?

We have overpaid our bill by

10.00
-9.90
credit .10

ten cents. This means we have a <u>credit</u> of $.10. Next we charge the suit: 24.40

we subtract the credit -.10

Balance 24.30

LET'S PUT IT INTO THE COMPUTER

1. To read and print last month's bill we use the data statement. We call the balance B, and make the first DATA last month's bill. We tell the computer to read and print it:

```
5 READ B
20 PRINT "BAL. FWD.",B
```

2. Next we tell the computer to make up a neat heading for our statement:

```
45 PRINT
50 PRINT "CHARGES", "BALANCE"
60 PRINT
```

Then we tell it to read the data, add up and print the balance as it goes along. We set up a loop to make sure it keeps doing this until it runs out of data:

```
100 READ X
200 LET B = B+X
300 PRINT X, B
400 GØTØ 100
500 DATA
999 END
```

All we have to do is put in the right numbers in the DATA line (line number 500),and we'll have the same program as we had to add up a bill.

3. To allow for payments by subtracting them, we can play a simple trick on the computer. JUST TYPE A MINUS SIGN IN FRONT OF EVERY PAYMENT. The faithful computer will do just what we tell it to do. If we tell it to add X(-10.00) to B(9.90) it will do this:

```
   9.90
 -10.00
 -  .10  ;
```

since adding a negative number (a payment) is the same as subtraction. If the negative number (payment) is larger than the positive number (balance), we get a negative number, which is a credit.

Since most people aren't used to thinking this way, we tell the computer to put its answers in plain English. To do this, we must put a "branch" in our program. We do this whenever a balance (B) or charge (X) is negative.

HOW TO PUT A BRANCH IN A PROGRAM

Let's start with last month's bill. If it is negative, we tell the computer to print it without the minus sign, and say "CREDIT":

```
10 IF B<0 THEN 40
40 PRINT "BAL. FWD.", -B ; "CREDIT"
```

Note how we do it. We use the expression on line 10, which really says,"If B is less than zero, then go to line 40."

(In BASIC there are several such expressions:

B<0 means "B is less than 0 (negative)";

B>0 means "B is greater than 0 (positive)";

B=0 means "B equals zero";

B<=0 means "B is either less than zero or zero";

B>=0 means "B is either greater than zero or zero";

or B<>0 means "B isn't equal to zero". These expressions can be used with numbers or variables other than 0, such as

B<2, B>X, B=Y, etc. A quantity like B or X is called a <u>variable</u>.)

We can put a like instruction later on in the program, in case our balance goes negative and becomes a credit:

```
350 IF B<0 THEN 380
380 PRINT -B; "CREDIT"
```

Finally, we can take care of payments, by typing them in as negative numbers:

```
210 IF X<0 THEN 320
320 PRINT -X: "PAID",
```

We now change line 300 and put in a few more lines, including the DATA statement in line 500.

```
300 PRINT X,
310 GØTØ 350
360 PRINT B
370 GØTØ 100
500 DATA 10.50,4.30,5.60,-10.50,-10.00,24.40
```

The reason for these changes can be best seen by asking the computer for a LIST of our program in correct order (if we type LISTNH, it types it without a heading. We type LISTNH whenever we want to punch a tape of our program)

```
LISTNH

5 READ B
10 IF B<0 THEN 40
20 PRINT "BAL. FWD.",B
30 GØTØ 45
40 PRINT "BAL. FWD.", -B ; "CREDIT"
45 PRINT
50 PRINT "CHARGES", "BALANCE"
60 PRINT
100 READ X
200 LET B = B+X
210 IF X<0 THEN 320
300 PRINT X,
310 GØTØ 350
320 PRINT -X; "PAID",
350 IF B<0 THEN 380
360 PRINT B
370 GØTØ 100
380 PRINT -B; "CREDIT"
400 GØTØ 100
500 DATA 10.50,4.30,5.60,-10.50,-10.00,24.40
999 END
```

It's not hard to understand how this program works. Just pretend to be a computer, and read the program line by line. (A flow chart on the next page will help you understand the program.)

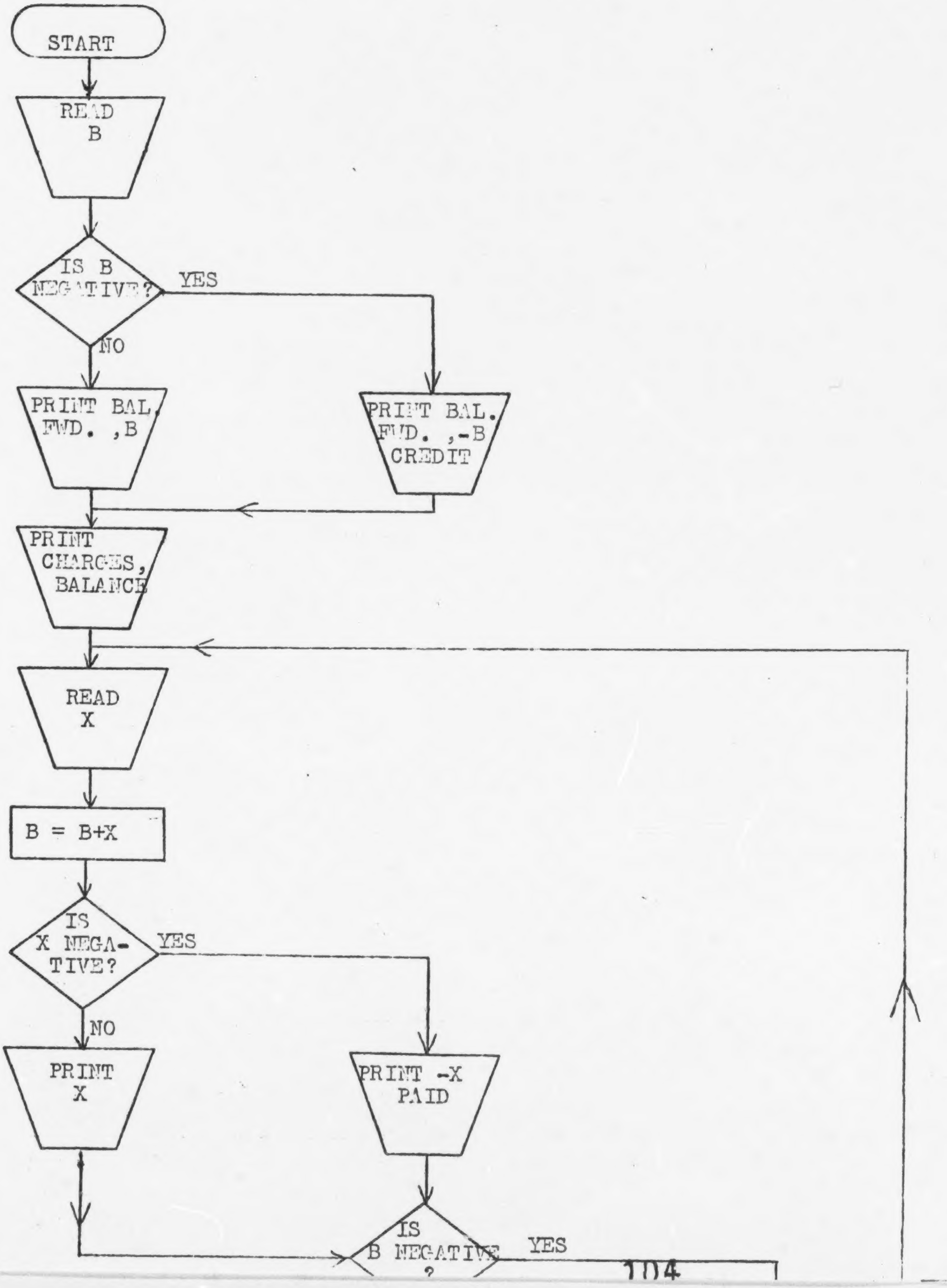
First read the balance, B. If B is negative, the computer goes to line 40, prints "BAL. FWD." The comma tells the computer to go over to the next zone (the page is divided into five zones of fifteen spaces each, in BASIC).

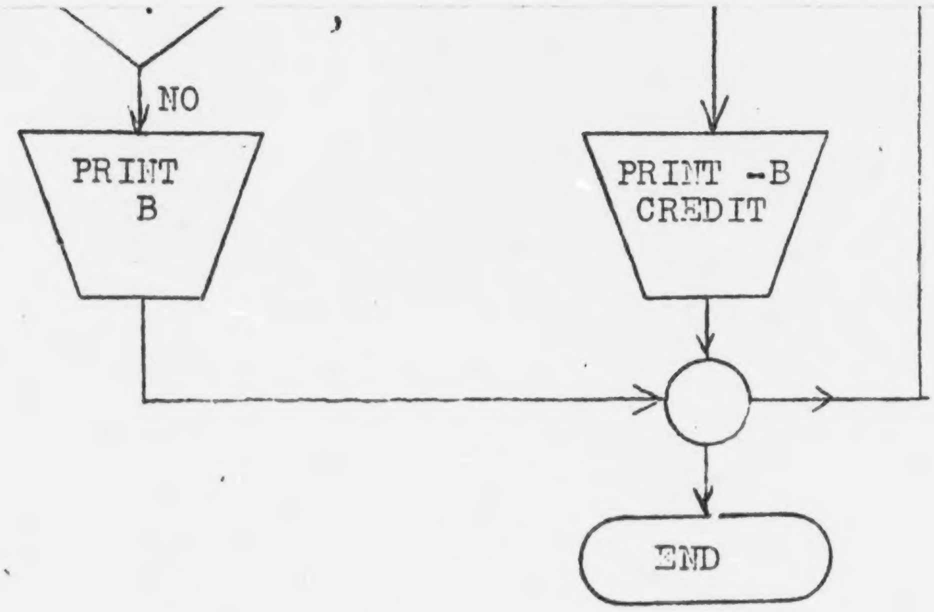
Then the computer prints -B, which will be positive. (The negative of a negative number is positive. -(-2)=2). Then, because of the ; mark, the computer prints "CREDIT" close by.

The computer goes on to the next part of the program (lines 45, 50 and 60) which leave a space, print a heading for the bill, and leave a space. This is one branch of the program.

The other branch is followed if B is zero or positive. If this is so, the computer ignores line 10 and goes on to the next

FLOW CHART FOR PREPARATION OF A MONTHLY BILL IN BASIC


START
READ B
IS B NEGATIVE?
YES
NO
PRINT BAL. FWD. ,B
PRINT BAL. FWD. ,-B CREDIT
PRINT CHARGES, BALANCE
READ X
B = B+X
IS X NEGA-TIVE?
YES
NO
PRINT X
PRINT -X PAID
IS B NEGATIVE ?
YES


NO
PRINT B
PRINT -B CREDIT
END

line 20. Line 30 is needed to make the computer skip over line 40, which now is the wrong branch of the program.

Lines 100 and 200 add the next charge or subtract the next payment. Line 210 sets up branches to take care of both cases. Line 300 or line 320 prints the amount of the charge or bill. The comma tells the computer "Go to the next zone, but stay on the same line. I want you to print something else there." Notice again the GØTØ statement in line 310, to make the computer skip line 320. Lines 350-380 are another pair of branches which print the balance (or credit), if B is positive (or negative). Line 400 takes the computer back to the beginning of the loop, line 500 contains the DATA, and line 999 the END.

The RUN for this program is shown below. Try to understand how the DATA were typed in for this bill. Then do the problems.

```
RUN

BILL          15:49   H   FRI 12/20/68

BAL. FWD.                10.5

CHARGES                 BALANCE

 4.3                     14.8
 5.6                     20.4
 10.5 PAID               9.9
 10   PAID               .1    CREDIT
 24.4                    24.3

ØUT ØF DATA  IN 100


USED         2.17 UNITS.
```

PROBLEMS:

1. A person has a credit of $10.00 at the beginning of the month. He buys a $5 book, half a dozen pairs of socks for $6.50 and a pair of pants for $10.00.Type up a new line of data for his account.

2. Make up a program to give the balance in a checking account. The balance will now be the amount of money on deposit. Let the first column stand for "CHECKS". If the person cashes checks for more money than is in his account, print the amount of money he owes the bank and print "ØVERDRAWN". (You can use the same program as BILL, just by changing a few words.)

3. Set up a program which will give the balance of a checking account for a bank which will not allow overdrafts. As soon as the account is overdrawn, do the same as in problem 2. Now, have the check "bounce" by refusing to cash it. Restore the balance to its old value before the check was cashed. (You will have to add a line to your solution of problem 2.)

An example of such a statement is shown below:

```
BAL. FWD.       10.05

CHECKS         BALANCE

 5 DEPØSIT      15.05
 10             5.05
 6.05           1.00 ØVERDRAWN
 6.05 DEPØSIT   5.05
```

DATA PROCESSING IN BASIC - A BANK STATEMENT

STRING VARIABLES

This is the third lesson in data processing. Here we make a few small changes in our program BILL to make it into BANK, a program to print out a bank statement for a checking account.

A bank statement should do the following jobs:

1-list the name of the person,
2-give the date of the statement,
3-give the previous balance,
4-subtract the amount of each check from the balance,
5-add each deposit to the balance,
6-handle overdrafts correctly,
7-end a statement without "ØUT ØF DATA" to mess it up.

We remember (from bitter experience) that, when we overdraw our account, the bank 'bounces" the check to prevent a negative balance.

STRING VARIABLES

Let's look at our program:

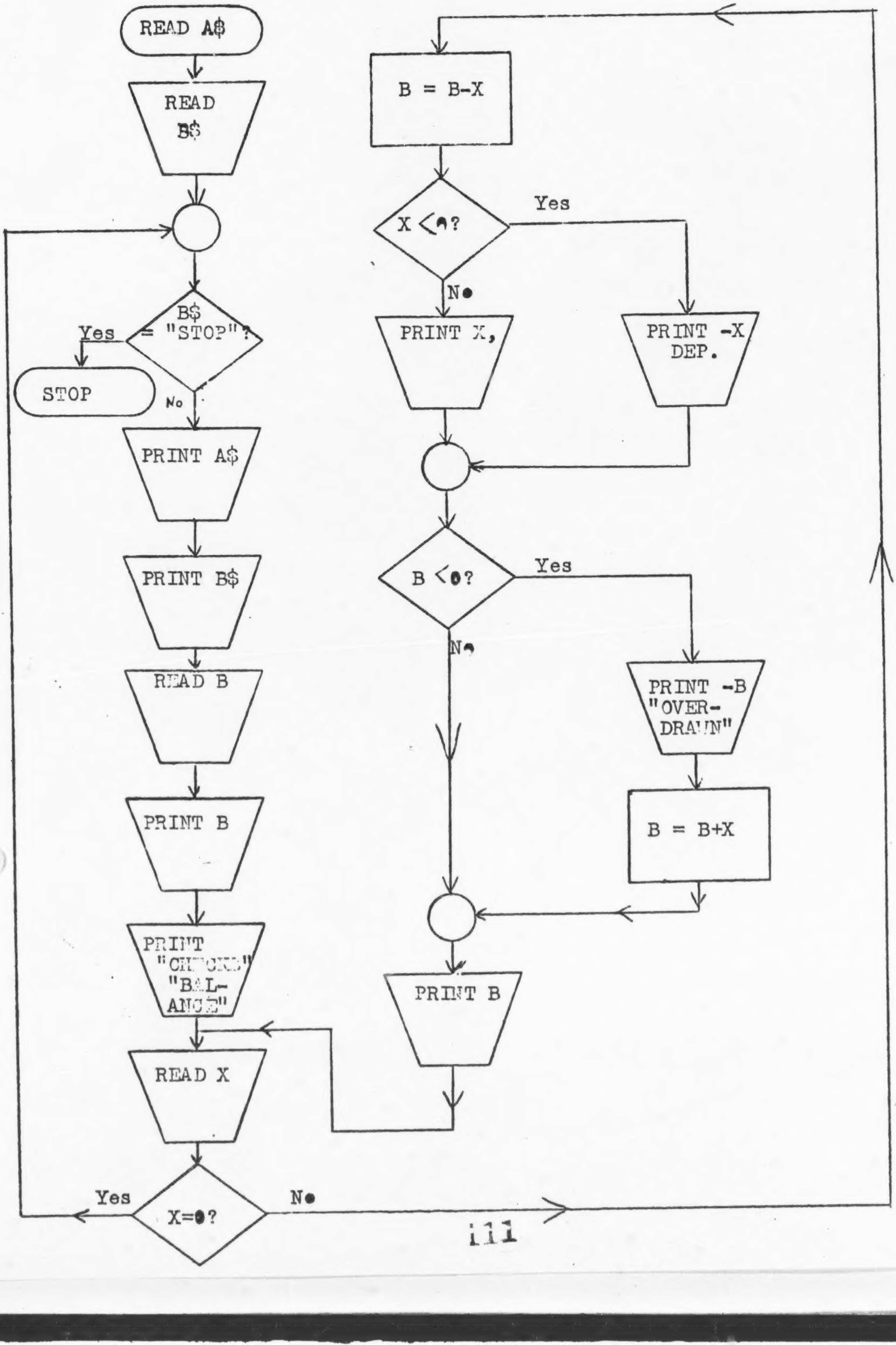
Line 1 brings a new kind of read statement:

```
1 READ A$
```

A letter with a $ mark after it stands for a "string variable." This is nothing more than a string of characters (letters, numbers, or even spaces). A$, in this case, stands for "MAR 5, 1969". Strings must follow these rules:

```
BANK       10:57    M  FRI 01/17/69

BANK STATEMENT AS ØF MAR 5,1969
ACCØUNT ØF WILLIAM SCHMIDT

BAL. FWD.           21.3

CHECKS              BALANCE

 11.2                10.1
 7.5   DEP.          17.6
 3.4                 14.2
 49    DEP.          63.2
 16                  47.2
 100                 52.8 ØVERDRAWN
                     47.2

BANK STATEMENT AS ØF MAR 5,1969
ACCØUNT ØF MARY JØNES

BAL. FWD.           1050.65

CHECKS              BALANCE

 225.5               825.15
 172.8               652.35
 262.75   DEP.       915.1
 80.64               834.46
 50.22               784.24
 10.11               774.13
 262.75   DEP.       1036.88
 200                 836.88
 131.7               705.18

USED       2.83 UNITS.
LISTNH

1 READ A$
2 PRINT
3 READ B$
4 IF B$ = "STØP" THEN 999
5 PRINT
6 PRINT "BANK STATEMENT AS ØF "; A$
7 PRINT "ACCØUNT ØF ";B$
8 PRINT
9 READ B
20 PRINT "BAL. FWD.",B
45 PRINT
```

```
50 PRINT "CHECKS","BALANCE"
60 PRINT
100 READ X
110 IF X= 0 THEN 2
200 LET B=B-X
210 IF X<0 THEN 320
300 PRINT X,
310 G0T0 350
320 PRINT -X; "DEP.",
350 IF B<0 THEN 380
360 PRINT B
370 G0T0 100
380 PRINT -B; "0VERDRAWN"
390 LET B = B+X
395 PRINT " ",
400 G0T0 360
500 DATA "MAR 5,1969"
501 DATA WILLIAM SCHMIDT
502 DATA 21.30,11.20,-7.50,3.40,-49.00,16.00,100.00,0
503 DATA MARY J0NES,1050.65,225.50,172.80,-262.75,80.64,50.22,10.11
504 DATA -262.75,200.00,131.70,0
998 DATA ST0P
999 END
```


READ A$
READ B$
B$ = "STOP"?
Yes
STOP
No
PRINT A$
PRINT B$
READ B
PRINT B
PRINT "CHECKING" "BAL-ANCE"
READ X
Yes
X=0?
No
B = B-X
X <0?
Yes
No
PRINT X,
PRINT -X DEP.
B <0?
Yes
No
PRINT -B "OVER-DRAWN"
B = B+X
PRINT B

1-The first part of A$ (or B$, C$, etc.) must be an alphabetical character, such as A, B, C, etc.
2-No commas can be in the string.
3-All spaces at the beginning or end are left out. Only spaces inside, as between "MAR" and "5", are kept.
4-If you want to use a comma, or want spaces at the beginning or end, put your string in quotation marks.
5-The string should be no more than 15 spaces long (on Mark I Basic. Mark II will take an entire line.)

The flow chart shows how our program works:

1-The computer reads the date, A$, in line 1.
2-Then it reads (line 3) and prints (line 7) B$, the name of the person holding the account.
3-The computer reads (line 9) and prints (line 20) the balance carried forward from last month.
4-Line 100 begins the loop to read and subtract (line 200) X, the amount of each check, from the balance, B. There are two branches to this loop. X is entered as a negative number (X < 0) for deposits. Line 210 begins a branch which prints -X (which is positive) as a deposit. If X is positive, line 300 begins the branch which handles the transaction as a check, and subtracts the correct amount from the balance.
5-In case a check is larger than the balance, the account would be overdrawn. This is handled in the branch in lines 380-400. Since banks don't allow overdrawn accounts, (negative balances) the check bounces (the bank won't cash it), and the account is put back to the balance before the check came in (lines 390-395).
6-The loop for each account is ended by a trick in line 110. Whenever X = 0, which can't be a deposit or a check, the computer leaves the loop, goes to line 2, and starts a new account.
7-The loop for the day's work is stopped in line 4. This prevents the message "ØUT ØF DATA". Line 998 is just left in the program at all times.

HOW THE DATA ARE PROCESSED

Notice how we can handle many accounts, just by putting in more lines of data. The program stays in the computer. Just the data are changed for each person.

The first piece of data (line 500) is A$, the date. This is changed only once each day.

The next piece of data is in line 501. It is B$, the name of the account.

The next data are the deposits and checks. These can go on and on. For example, in Mary Jones' account (lines 503-504), it takes more than one line of data to put in the information.

By now you may see the power of the computer to process data. We just type in the right names and numbers, in the proper order, for each account. After we are finished, we feed in the data, and the computer does the rest. In actual banks, the checks (and deposit slips) carry the deposit number in magnetic ink. A data processing machine operator types on the amount of the deposit or check in magnetic ink. The computer sorts out the checks. The amounts are automatically put on magnetic tape. At the end of the month, the data from the tape are read off by the computer and a statement is made up. This program, BANK, does the last step of the data processing. In a similar fashion, computers can make up payrolls.

PROBLEMS:

1- John Doe had an initial balance of $230.56. He deposited 112.50 on each of his four paydays. He gava his wife four checks for $75.00. He also paid bills for electricity, $10.52; for gas, $2.35, and for his telephone, $7.85. Make up a line or two of data for his account and add it to the program. Run it off.

2- This problem is harder. Write a program to make a statement for an account. Enter deposits in the first column, withdrawals in the second column, and balance in the third. Otherwise, keep it the same as before. Use your program to make statements for William Schmidt's and Mary Jones' accounts.

DENSE, A Computer Program for Physical Science,
written by William Lichten

A common exercise in physical science laboratories is to measure the length, L, width, W, and height, H, of a block of metal. The mass, M, is found by weighing and the density, D, is calculated. This computer program does the numerical work by substituting these values in the formulae:

$$D = \frac{M}{V}\ , \quad \text{where}$$

$$V = L \times W \times H$$

The result is given to six digits. This is more than the correct number of significant figures, which is equal to the smallest number of digits in any of the numbers L, W, H or M.

Note that lines 200 and 210 are the heart of the programs. A simple three line program could be written to solve this problem. A sample run is given after the word "run".

```
10 PRINT "THIS IS A CØMPUTER PRØGRAM TØ PRØCESS LABØRATØRY DATA."
20 PRINT"I'LL FIND THE DENSITY ØF A RECTANGULAR SØLID."
45 PRINT
50 PRINT"WHEN I ASK YØU TØ,GIVE THE LENGTH, WIDTH, AND HEIGHT ØF YØUR"
60 PRINT"RECTANGULAR SØLID IN CENTIMETERS.SEPARATE THE THREE NUMBERS BY"
70 PRINT"CØMMAS. THEN PRESS THE 'RETURN' KEY AFTER EACH ANSWER YØU GIVE"
80 PRINT "THIS IS IMPØRTANT. FØR EXAMPLE: 8.88, 2.53, .61 'RETURN' "
85 PRINT
90 PRINT "WHAT ARE YØUR THREE NUMBERS";
95 INPUT L(1),L(2),L(3)
96 PRINT"GIVE THE MASS ØF YØUR BLØCK,TYPE A CØMMA,THEN TYPE THE UNITS"
97 PRINT"(IN GRAMS, BEADS, ETC.). FØR EXAMPLE: 37, GRAMS 'RETURN' "
98 PRINT
100 PRINT"WHAT IS THE MASS, UNITS ØF MASS ØF YØUR BLØCK";
110 INPUT L(4),B$
115PRINT
200 LET V = L(1)*L(2)*L(3)
210 LET D = L(4)/V
220 PRINT "THE VØLUME IS EQUAL TO "; V ;" CUBIC CENTIMETERS."
230 PRINT"THE DENSITY IS EQUAL TØ ";D;" ";B$;" /CC. THIS RESULT HAS TØØ"
240 PRINT"MANY DIGITS. RØUND IT ØFF. LEAVE ØNLY SIGNIFICANT FIGURES."
400 PRINT "DØ YØU WANT TØ DØ ANØTHER (TYPE YES ØR NØ)";
410 INPUT A$
420 IF A$ = "YES" THEN 90
430 IF A$ = "NØ" THEN 500
440 PRINT "I DIDN'T UNDERSTAND YØU. PLEASE FØLLØW INSTRUCTIØNS."
450 GØTØ 400
500 PRINT "GØØDBYE. CØME AGAIN. NØW SAY GØØDBYE TØ ME."
999 END
```

```
RUN
```

```
DENSE      11:17   M   WED 10/30/68

THIS IS A CØMPUTER PRØGRAM TØ PRØCESS LABØRATØRY DATA.
I'LL FIND THE DENSITY ØF A RECTANGULAR SØLID.

WHEN I ASK YØU TØ, GIVE THE LENGTH, WIDTH, AND HEIGHT ØF YØUR
RECTANGULAR SØLID IN CENTIMETERS.SEPARATE THE THREE NUMBERS BY
CØMMAS. THEN PRESS THE 'RETURN' KEY AFTER EACH ANSWER YØU GIVE
THIS IS IMPØRTANT. FØR EXAMPLE: 8.88, 2.53, .61 'RETURN'

WHAT ARE YØUR THREE NUMBERS? 1.2,1.2,1.2
GIVE THE MASS ØF YØUR BLØCK,TYPE A CØMMA, THEN TYPE THE UNITS
[IN GRAMS, BEADS, ETC.]. FØR EXAMPLE: 37, GRAMS 'RETURN'

WHAT IS THE MASS, UNITS ØF MASS ØF YØUR BLØCK? 28.2

INPUT DATA NØT IN CØRRECT FØRMAT. RETYPE IT.
? 28.2,BEADS

THE VØLUME IS EQUAL TØ 1.728   CUBIC CENTIMETERS.
THE DENSITY IS EQUAL TØ  16.3194  BEADS/CC. THIS RESULT HAS TØØ
MANY DIGITS. RØUND IT ØFF. LEAVE ØNLY SIGNIFICANT FIGURES.
DØ YØU WANT TØ DØ ANØTHER [TYPE YES ØR NØ]? YEA
I DIDN'T UNDERSTAND YØU. PLEASE FØLLØW INSTRUCTIØNS.
DØ YØU WANT TØ DØ ANØTHER [TYPE YES ØR NØ]? YES
WHAT ARE YØUR THREE NUMBERS? 7.5,2.5,.6
GIVE THE MASS ØF YØUR BLØCK,TYPE A CØMMA, THEN TYPE THE UNITS
[IN GRAMS, BEADS, ETC.]. FØR EXAMPLE: 37, GRAMS 'RETURN'

WHAT IS THE MASS, UNITS ØF MASS ØF YØUR BLØCK? 169.7,BEADS

THE VØLUME IS EQUAL TØ  11.25    CUBIC CENTIMETERS.
THE DENSITY IS EQUAL TØ  15.0844  BEADS/CC. THIS RESULT HAS TØØ
MANY DIGITS. RØUND IT ØFF. LEAVE ØNLY SIGNIFICANT FIGURES.
DØ YØU WANT TØ DØ ANØTHER [TYPE YES ØR NØ]? NØ
GØØDBYE. CØME AGAIN. NØW SAY GØØDBYE TØ ME.



RAN     14.83 SEC
GØØDBYE

*** ØFF AT 11:28  ELAPSED TERMINAL TIME =     14 MIN.
```

Note that this was conducted in an actual classroom. The data was typed in by the pupils themselves. The students, who were in an "average" section (0 - 50 percentile) had no preparation, other than the fact that they had the data from a previous experiment.

This experiment uses two pieces of aluminum (density 2.7 grams/cc or 13.5 beads/cc), a cube and a long slab. The pieces are weighed, measured and the densities are calculated. The data and computer results are shown in the table

	Length	Width	Height	Volume	Mass	Density = Mass/Volume
Cube	1.2cm	1.2cm	1.2cm	1.728cm^3	28.2 Beads	16.3194 Beads/CC
Slab	7.5cm	2.5cm	.6cm	11.25cm^3	169.7 Beads	15.0844 Beads/CC

Within the accuracy of this type of ninth grade, physical science class, the agreement between the two densities is reasonably good. This indicates that both the cube and slab have the same density, and may be the same material.

FACTØR, A Program To Find The
Prime Factors Of An Integer

This program finds the prime factors of an integer, X, by dividing by an integer Y. Y starts with the value 2. If X/2 is not integral, Y is set equal to 3, then 4, etc. It stops as soon as the quotient Z = X/Y is an integer. The computer then prints the prime factor Y. (Y must be prime. Otherwise, one of its factors would have been reached.) Then the computer replaces X with the new value equal to the old value of the quotient Z. Y is kept equal to its last value. The process is then repeated until one of two conditions happens:

1. If Z = 1, all the factors are found. The last factor Y is printed and the program is complete.

2. If $Y > \sqrt{X}$, there is no point in trying larger factors. For example, in finding the factors of 17, no factor of 17 can be equal to, or greater than $\sqrt{17}$. All numbers, 2, 3, 4, and 5 have been tried and none is a factor. Suppose any number greater than 5 (which is greater than $\sqrt{17}$) were a factor, for example 6. Then 17/6 would be an integer which is a factor. But, since $6 > \sqrt{17}$, then $17/6 < \sqrt{17}$. However, the computer has shown that no integer $< \sqrt{17}$ was a factor. Therefore, our assumption is contrary to fact, and no integer larger than $\sqrt{17}$ can be a prime factor of 17.

The computer therefore stops whenever $Y > \sqrt{X}$ and prints X as a prime factor. This saves a great deal of computer time when large prime factors, such as 25,391 are entered. Without this feature, the computer has to try 25,390 integers. With it, only 159 possible prime factors need be tried.

The program does not avoid trying non-prime integers, such as 4, 6, 8, etc. However, unless numbers of the order of 10^6 or larger are going to be entered, the time taken to solve this problem is small compared to the time to set it up.

A flow chart diagram is shown, along with the list and a classroom run. In trying to follow this flow chart, it is helpful to take an actual number, such as 14, and follow through the steps.

The classroom demonstration was for several first year algebra classes that were on factoring. It seemed most effective with the less advanced students.

```
FACTØR    14:01     M   TUE 11/26/68

THIS PRØGRAM WILL GIVE THE PRIME FACTØRS ØF ANY
INTEGER.  YØU GIVE THE INTEGER, X, AND I WILL GIVE
THE FACTØRS. WHEN I ASK YØU TØ, TYPE X AND THEN 'RETURN'
FØR EXAMPLE:
WHAT IS YØUR VALUE FØR X? 24 'RETURN'

WHAT IS YØUR VALUE FØR X? 18

THE PRIME FACTØRS ØF 18  ARE 2  3  3

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? YES

WHAT IS YØUR VALUE FØR X? 9

THE PRIME FACTØRS ØF 9 ARE 3  3

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? YES

WHAT IS YØUR VALUE FØR X? 168000

THE PRIME FACTØRS ØF 168000 ARE 2 2 2 2 2 2 3 5 5 5 7

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? YES

WHAT IS YØUR VALUE FØR X? 3768000

THE PRIME FACTØRS ØF 3768000 ARE 2 2 2 2 2 2 3 5 5 5 157

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? YES

WHAT IS YØUR VALUE FØR X?  157

THE PRIME FACTØRS ØF 157  ARE 157

DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)? STØP

RAN     16.00 SEC
```

```
LISTNH
10 PRINT "THIS PRØGRAM WILL GIVE THE PRIME FACTØRS ØF ANY "
20 PRINT "INTEGER. YØU GIVE THE  INTEGER, X, AND  I WILL GIVE"
30 PRINT "THE FACTØRS. WHEN I ASK YØU TØ, TYPE X AND THEN 'RETURN'"
40 PRINT "FOR EXAMPLE:"
50 PRINT "WHAT IS YØUR VALUE FØR X? 24 'RETURN' "
100 PRINT
105 PRINT "WHAT IS YØUR VALUE FØR X";
110 INPUT X
115 PRINT
1[illegible] PRINT "THE PRIME FACTØRS ØF"; X ; "ARE";
120 LET Y = 2
125 IF Y <= SØR(X) THEN 130
126 LET Y = X
130 LET Z=X/Y
140 IF Z-INT(Z)=0 THEN 150
141 LET Y=Y+1
142 GØTØ 125
150 IF Z=1 THEN 160
151 PRINT Y;
152 LET X=Z
[illegible]53 GØTØ 125
160 PRINT Y;
165 PRINT
168 PRINT
170 PRINT "DØ YØU WANT TØ DØ ANØTHER (ANSWER YES ØR NØ)";
180 INPUT A$
185 PRINT
190 IF A$ = "YES" THEN 100
200 IF A$ = "NØ" THEN 220
210 PRINT "SØRRY, I DIDN'T UNDERSTAND YØU.  PLEASE FØLLØW DIRECTIØNS."
215 GØTØ 170
2[illegible] PRINT "IT'S BEEN A PLEASURE TØ SERVE YØU. CØME AGAIN. NØW,"
230 PRINT "SAY GØØDBYE TØ ME. GØØDBYE."
999 END
```

FUN, A Function Machine

FUN is a simple program, invented by David Howell, Chairman of the Lee High School Mathematics department. In it the computer acts as a "function machine." Given a value of x, it prints a corresponding value of y. The user can change the function by calling up the program and retyping line 100 to his specification. For example, if he wanted y= 2x+3, he could type:

```
100   DEF FNA(x)= 2*x+3
RUN
```

A list and sample run are attached.

```
WLLFUN    12:11    M  WED 06/25/69

HELLO THERE, MATH WIZARDS. I'LL BE A FUNCTION MACHINE.
YOU GIVE ME AN 'X', AND I'LL GIVE YOU 'Y'.
SEE IF YOU CAN GUESS WHAT FUNCTION 'Y' IS.


WHAT IS YOUR VALUE OF 'X', PLEASE? 101

WHEN X = 101    Y =-.01


MORE (YES OR NO)? PERHAPS
PLEASE ANSWER YES OR NO.
MORE (YES OR NO)? NO


I'VE ENJOYED BEING YOUR FUNCTION MACHINE. COME AGAIN. GOODBYE.




USED      7.00 UNITS
LISTNH

100 DEF FNA(X) = 1/(1-X)
110 PRINT "HELLO THERE, MATH WIZARDS. I'LL BE A FUNCTION MACHINE."
120 PRINT"YOU GIVE ME AN 'X', AND I'LL GIVE YOU 'Y'."
130 PRINT"SEE IF YOU CAN GUESS WHAT FUNCTION 'Y' IS."
140 PRINT
150 PRINT
160 PRINT"WHAT IS YOUR VALUE OF 'X', PLEASE";
170 INPUT X
180 IF ABS(FNA(X))>1E76 THEN 220
190 PRINT
200 PRINT"WHEN X =";X,"Y =";FNA(X)
210 GO TO 250
220 PRINT"(THIS MEANS THAT 'Y' HAS A ZERO IN ITS DENOMINATOR."
230 PRINT"THAT IS A NO-NO. THE COMPUTER CAN'T FIGURE OUT WHAT"
240 PRINT"'Y' REALLY IS, AND TELLS US SO.)"
250 PRINT
260 PRINT
270 PRINT "MORE (YES OR NO)";
280 INPUT AS
290 IF AS = "YES" THEN 160
300 IF AS="NO" THEN 330
310 PRINT "PLEASE ANSWER YES OR NO."
320 GOTO 270
330 PRINT
340 PRINT
350 PRINT "I'VE ENJOYED BEING YOUR FUNCTION MACHINE. COME AGAIN. GOODBYE"
360 END
```

Function Programs:

MORFUN, a Function of One Variable

XY-FUN, A Function of Two Variables

FUN, a function machine for single variables, was improved for class-room use by these two programs. MORFUN allowed the instructor to use any one of a set of eighteen functions in rapid sucession in class, without having to revise the program "on line". This has the obvious advantage of not "spilling the bean " in front of the class when a new function is used. Also, it requires less skill on the part of the teacher. All the teacher needs is the list of functions, which is given in readily readable form in the list of the program. The functions are chosen by giving two numbers. The first number specifies the level of difficulty and the second number gives a choice of different functions within that level. In the run that is attached, the first function (1,1) was Y = X + 3, a very easy one. The class (an advanced group of students) got that one after only two times. The second function (2,4) was Y = 1 + 1/X This proved quite hard, until the class had the good fortune of choosing X = 0. This revealed that the expression had an expression of the form a/X present, and the class soon realized what the expression was.

The list of MORFUN follows the sample run. It is worth pointing out functions 2,1 (absolute value of x), 3,2 (the half integer nearest to x), 4,2 (the integer nearest to x) 4,3 (the "square wave" function), 4,4 and 4,5 (prime integers nearest to x) are given instantly by the computer and reveal its great power and versatility.

The prime integer functions are obtained by a subroutine which is in lines 720-810, which is based on FACTØR, the program listed elsewhere in this library.

31

```
?RUN

MORFUN        13:38       07/09/69

HELLO THERE, MATH WIZARDS. I'LL BE A FUNCTION MACHINE.
FIRST, WILL THE INSTRUCTOR PLEASE PUT IN THE CODE NUMBER
FOR THE DESIRED FUNCTION? 1,1

NOW IT'S YOUR TURN TO HAVE FUN, STUDENTS.
YOU GIVE ME AN 'X', AND I'LL GIVE YOU 'Y'.
SEE IF YOU CAN GUESS WHAT FUNCTION 'Y' IS.


WHAT IS YOUR VALUE OF 'X', PLEASE? 2


WHEN X = 2     Y = 5


MORE (YES OR NO)? YES
WHAT IS YOUR VALUE OF 'X', PLEASE? 3


WHEN X = 3     Y = 6


MORE (YES OR NO)? NO


WOULD THE INSTRUCTOR LIKE TO TRY ANOTHER FUNCTION? YES
PLEASE ENTER THE CODE NUMBER FOR THE DESIRED FUNCTION? 2,4

NOW IT'S YOUR TURN TO HAVE FUN, STUDENTS.
YOU GIVE ME AN 'X', AND I'LL GIVE YOU 'Y'.
SEE IF YOU CAN GUESS WHAT FUNCTION 'Y' IS.


WHAT IS YOUR VALUE OF 'X', PLEASE? 2


WHEN X = 2     Y = 1.5
```

```
MORE (YES OR NO)? YES
WHAT IS YOUR VALUE OF 'X', PLEASE? 4

WHEN X = 4      Y = 1.25

MORE (YES OR NO)? YES
WHAT IS YOUR VALUE OF 'X', PLEASE? 0

DIVISION BY ZERO IN 420
(THIS MEANS THAT 'Y' HAS A ZERO IN ITS DENOMINATOR.
THAT IS A NO-NO. THE COMPUTER CAN'T FIGURE OUT WHAT
'Y' REALLY IS, AND TELLS US SO.)

MORE (YES OR NO)? YES
WHAT IS YOUR VALUE OF 'X', PLEASE? .25

WHEN X = 0.25  Y = 5

MORE (YES OR NO)? NO

WOULD THE INSTRUCTOR LIKE TO TRY ANOTHER FUNCTION? NO

I'VE ENJOYED BEING YOUR FUNCTION MACHINE. COME AGAIN. GOODBYE.

USED     .39 UNITS

MORFUN
```

```
100 PRINT"HELLO THERE, MATH WIZARDS. I'LL BE A FUNCTION MACHINE."
110 PRINT"FIRST, WILL THE INSTRUCTOR PLEASE PUT IN THE CODE NUMBER"
120 PRINT"FOR THE DESIRED FUNCTION";
130 INPUT A,B
140 PRINT
150 PRINT"NOW IT'S YOUR TURN TO HAVE FUN, STUDENTS."
160 PRINT"YOU GIVE ME AN 'X', AND I'LL GIVE YOU 'Y'."
170 PRINT"SEE IF YOU CAN GUESS WHAT FUNCTION 'Y' IS."
180 PRINT
190 PRINT
200 PRINT"WHAT IS YOUR VALUE OF 'X', PLEASE";
210 INPUT X
220 PRINT
230 ON A GOTO 240,350,460,570                      'FUNCTIONS:
240 ON B GOTO 250,270,290,310,330
250 LET
260 GOTO 810                       Y=X+3                 'CODE 1,1
270 LET
280 GOTO 810                       Y=2*X                 'CODE 1,2
290 LET
300 GOTO 810                       Y=X-1                 'CODE 1,3
310 LET
320 GOTO 810                       Y=2*X+2               'CODE 1,4
```

```
330 LET                                      Y=X/2             'CODE 1,5
340 GOTO 810
350 ON B GOTO 360,380,400,420,440
360 LET                                      Y=ABS(X)          'CODE 2,1
370 GOTO 810
380 LET                                      Y=3               'CODE 2,2
390 GOTO 810
400 LET                                      Y=X↑2             'CODE 2,3
410 GOTO 810
420 LET                                      Y=1+1/X           'CODE 2,4
430 GOTO 810
440 LET                                      Y=5*X/2           'CODE 2,5
450 GOTO 810
460 ON B GOTO 470,490,510,530,550
470 LET                                      Y=1/(1-X)         'CODE 3,1
480 GOTO 810
490 LET                                      Y=INT(X-.5)+.5 'CODE 3,2
500 GOTO 810
510 LET                                      Y=(X+1)/(X-1)     'CODE 3,3
520 GOTO 810
530 LET                                      Y=X↑2+2*X+2       'CODE 3,4
540 GOTO 810
550 LET                                      Y=9-X↑2           'CODE 3,5
560 GOTO 810
570 ON B GOTO 580,600,620,670,700
580 LET                                      Y=-INT(-X)        'CODE 4,1
590 GOTO 810
600 LET                                      Y=INT(X+.5)       'CODE 4,2
610 GOTO 810
620 IF X=0 THEN 650                         'Y= 1 IF X>0
630 LET Y=ABS(X)/X                          'Y= 0 IF X=0      CODE 4,3
640 GOTO 810                                'Y=-1 IF X<0
650 LET Y=0
660 GOTO 810
670 LET B=1                                 'Y=PREVIOUS PRIMECODE 4,4
680 LET A=-1
690 GOTO 720
700 LET A=1                                 'Y=NEXT PRIME     CODE 4,5
710 LET B=2*X
720 IF X<1 THEN 800
730 FOR I=INT(X)+A TO B STEP A
740 FOR J=2 TO SQR(I)
750 IF I/J=INT(I/J) THEN 790
760 NEXT J
770 LET Y=I
780 GOTO 810
790 NEXT I
800 PRINT"THIS FUNCTION IS NOT DEFINED FOR NUMBERS UNDER 2."
810 IF Y>2↑30 THEN 850
820 PRINT
830 PRINT"WHEN X =";X,"Y =";Y
840 GOTO 880
850 PRINT"(THIS MEANS THAT 'Y' HAS A ZERO IN ITS DENOMINATOR."
860 PRINT"THAT IS A NO-NO. THE COMPUTER CAN'T FIGURE OUT WHAT"
870 PRINT"'Y' REALLY IS, AND TELLS US SO.)"
880 PRINT
890 PRINT
900 PRINT "MORE (YES OR NO)";
910 INPUT A$
920 IF A$ = "YES" THEN 200
```

```
930 IF A$="NO" THEN 960
940 PRINT "PLEASE ANSWER YES OR NO."
950 GOTO 900
960 PRINT
970 PRINT
980 PRINT"WOULD THE INSTRUCTOR LIKE TO TRY ANOTHER FUNCTION";
990 INPUT B$
1000 IF B$="NO" THEN 1060
1010 IF B$="YES" THEN 1040
1020 PRINT"COME ON, TEACH.  YOU OUGHT TO KNOW BETTER.  YES OR NO";
1030 GOTO 990
1040 PRINT"PLEASE ENTER THE CODE NUMBER FOR THE DESIRED FUNCTION";
1050 GOTO 130
1060 PRINT
1070 PRINT
1080 PRINT "I'VE ENJOYED BEING YOUR FUNCTION MACHINE. COME AGAIN. GOODBY
1090 END
```

• XY-FUN

XY-FUN is an extension of MORFUN to a function of two variables. The teacher uses the same method or presentation as in MORFUN. In the sample run, the user first chose 2,2 (z = x/y). The second choice was 3,5 (z is the greatest common factor of X and Y).

It can be seen from the list that the use of two variables allows a much more imaginative choice of relationships between two variables. Simple relationships, such as sum, product, quotient, difference are present. More complicated functions, such as 2,6 (z= $\sqrt{x^2 + y^2}$) also are represented. Finally, the greatest common factor and least common multiple are represented, by use of the program L.C.M. (from the program library) as a subroutine in lines 658-700.

These programs are written for use on the form of BASIC used on the GE-635 computer, which is available at Dartmouth College or on the GE Mark II time-sharing service.

These programs were written by a high-school student, Leonard Horwitz.

5

```
RUN

XY-FUN        11:15       07/03/69

DØUBLE, DØUBLE, TØIL AND TRØUBLE...

SØ, YØU MATH WIZARDS HAVE MASTERED 'MØRFUN', EH?  WELL, WE
FUNCTIØN MACHINES DØN'T GIVE UP SØ SØØN.  SØ NØW, LET'S TRY
FUNCTIØNS ØF TWØ VARIABLES.  GUESS WHAT FUNCTIØN PRØDUCES Z.

WILL THE TEACHER PLEASE ENTER THE CØDE NUMBER ØF THE DESIRED
FUNCTIØN ØF TWØ VARIABLES? 2,2


NØW, STUDENTS, ENTER THE X AND Y VALUES WHEN I ASK FØR
THEM.  MAKE SURE THEY ARE INTEGERS ØR DECIMALS (AS .3, -1;
I HATE FRACTIØNS, TØØ!).  ALSØ, SEPARATE VALUES BY A CØMMA.

YØUR VALUES ØF X AND Y, PLEASE? 4,2
```

```
WHEN X = 4      AND Y = 2 ,     Z = 2


MORE (YES OR NO)? YES
YOUR VALUES OF X AND Y, PLEASE? 9,6

WHEN X = 9      AND Y = 6 ,     Z = 1.5


MORE (YES OR NO)? YES
YOUR VALUES OF X AND Y, PLEASE? 3,0

DIVISION BY ZERO IN 380
(THIS MEANS THAT Z HAS A ZERO IN ITS DENOMINATOR.
Z IS TOO LARGE FOR THE COMPUTER TO FIGURE OUT.)


MORE (YES OR NO)? YES
YOUR VALUES OF X AND Y, PLEASE? 0,3

WHEN X = 0      AND Y = 3 ,     Z = 0


MORE (YES OR NO)? NO


WOULD THE CLASS LIKE TO TRY ANOTHER FUNCTION? YES
WILL THE TEACHER PLEASE ENTER THE CODE NUMBER OF THE DESIRED
FUNCTION OF TWO VARIABLES? 3,5


NOW, STUDENTS, ENTER THE X AND Y VALUES WHEN I ASK FOR
THEM.  MAKE SURE THEY ARE INTEGERS OR DECIMALS (AS .3, -1;
I HATE FRACTIONS, TOO!).  ALSO, SEPARATE VALUES BY A COMMA.

YOUR VALUES OF X AND Y, PLEASE? 200,240

WHEN X = 200    AND Y = 240 ,   Z = 40


MORE (YES OR NO)? YES
YOUR VALUES OF X AND Y, PLEASE? 1684,3726

WHEN X = 1684   AND Y = 3726 , Z = 2


MORE (YES OR NO)? NO


WOULD THE CLASS LIKE TO TRY ANOTHER FUNCTION? NO


I'VE ENJOYED BEING YOUR FUNCTION MACHINE.  COME AGAIN.  BYE.


USED     .39 UNITS
```

C

XY-FUN

```
10PRINT"DOUBLE, DOUBLE, TOIL AND TROUBLE..."
20 PRINT
30 PRINT"SO, YOU MATH WIZARDS HAVE MASTERED 'MORFUN', EH?  WELL, WE"
40 PRINT"FUNCTION MACHINES DON'T GIVE UP SO SOON.  SO NOW, LET'S TRY"
50 PRINT"FUNCTIONS OF TWO VARIABLES.  GUESS WHAT FUNCTION PRODUCES Z."
60 PRINT
70 PRINT"WILL THE TEACHER PLEASE ENTER THE CODE NUMBER OF THE DESIRED"
80 PRINT"FUNCTION OF TWO VARIABLES";
110 INPUT A,B
120 PRINT
130 PRINT
140 PRINT"NOW, STUDENTS, ENTER THE X AND Y VALUES WHEN I ASK FOR"
150 PRINT"THEM.  MAKE SURE THEY ARE INTEGERS OR DECIMALS (AS .3, -1;"
160 PRINT"I HATE FRACTIONS, TOO!).  ALSO, SEPARATE VALUES BY A COMMA."
180 PRINT
190 PRINT"YOUR VALUES OF X AND Y, PLEASE";
200 INPUT X,Y                                    'FUNCTIONS:
205 PRINT
210 ON A GOTO 220,350,430
220 ON B GOTO 230,250,270,290,310,330
230 LET                                Z=X+Y              'CODE 1,1
240 GOTO 730
250 LET                                Z=X-Y              'CODE 1,2
260 GOTO 730
270 LET                                Z=Y-X              'CODE 1,3
280 GOTO 730
290 LET                                Z=ABS(X-Y)         'CODE 1,4
300 GOTO 730
310 LET                                Z=X                'CODE 1,5
320 GOTO 730
330 LET                                Z=Y                'CODE 1,6
340 GOTO 730
350 ON B GOTO 360,380,400,420,440,460
360 LET                                Z=X*Y              'CODE 2,1
370 GOTO 730
380 LET                                Z=X/Y              'CODE 2,2
390 GOTO 730
400 LET                                Z=Y/X              'CODE 2,3
410 GOTO 730
420 LET                                Z=X↑Y              'CODE 2,4
430 GOTO 730
440 LET                                Z=Y↑X              'CODE 2,5
450 GOTO 730
460 LET                                Z=SQR(X↑2+Y↑2)     'CODE 2,6
470 GOTO 730
480 ON B GOTO 490,520,550,570,590,620
490 IF X>Y THEN 530                   'Z=GREATER           CODE 3,1
```

```
500 LET Z=Y
510 GOTO 730
520 IF X>Y THEN 500                             'Z=LESSER          CODE 3,2
530 LET Z=X
540 GOTO 730
550 LET                                          Z=(X+Y)/2        'CODE 3,3
560 GOTO 730
570 LET                                          Z=SQR(X*Y)       'CODE 3,4
580 GOTO 730
590 GOSUB 650                                   'Z=G.C.F.          CODE 3,5
600 LET Z=V
610 GOTO 730
620 GOSUB 650                                   'Z=L.C.M.          CODE 3,6
630 LET Z=X*Y/V
640 GOTO 730
650 IF X*Y<1 THEN 654
652 IF X*Y=ABS(INT(X))*INT(Y) THEN 658
654 PRINT"SORRY.  THIS FUNCTION IS ONLY GOOD FOR POSITIVE INTEGERS."
656 GOTO 800
658 LET U=Y
660 LET V=X
670 LET Q=INT(V/U)
680 LET R=V-Q*U
690 LET V=U
700 LET U=R
710 IF R>0 THEN 670
720 RETURN
730 IF ABS(Z)>2↑30 THEN 770
750 PRINT"WHEN X ="; X,"AND Y =";Y;",","Z =";Z
760 GOTO 790
770 PRINT"(THIS MEANS THAT Z HAS A ZERO IN ITS DENOMINATOR."
780 PRINT"Z IS TOO LARGE FOR THE COMPUTER TO FIGURE OUT.)"
790 PRINT
800 PRINT
810 PRINT"MORE (YES OR NO)";
820 INPUT C$
830 IF C$="YES" THEN 190
840 IF C$="NO" THEN 870
850 PRINT"PLEASE ANSWER EITHER YES OR NO."
860 GOTO 810
870 PRINT
880 PRINT
890 PRINT"WOULD THE CLASS LIKE TO TRY ANOTHER FUNCTION";
900 INPUT D$
910 IF D$="YES" THEN 70
920 IF D$="NO" THEN 950
930 PRINT"PLEASE ANSWER EITHER YES OR NO."
940 GOTO 890
950 PRINT
960 PRINT
970 PRINT"I'VE ENJOYED BEING YOUR FUNCTION MACHINE.  COME AGAIN.  BYE."
999 END
```

HITHAR, A Program to illustrate subroutines.

The following program is self-explanatory;

```
RUN.


HITHAR    13:41    M  THU 03/20/69

*********************************************************************


HELL0 THERE, Y0U SCH0LARS 0F THE DIGITAL C0MPUTER AND 0F ALL
ITS W0NDERS. THIS PR0GRAM IS T0 SH0W 0NE 0F THEM, NAMELY,
THE G0SUB C0MMAND. IT'S REALLY GREAT.



*********************************************************************


N0TICE H0W WE CAN USE THE G0SUB C0MMAND T0 D0 THE SAME THING
AT DIFFERENT PARTS 0F THE SAME PR0GRAM.


Y0U C0ULD MAKE UP A REALLY NIFTY BANK STATEMENT BY USING THE
G0SUB C0MMAND T0 SPRUCE IT UP WITH SPACES AND ASTERISKS ALL
0VER THE PLACE.


G0SUB 950 MAKES A TRIPLE SPACE. G0SUB 900 MAKES A R0W 0F *'S.
Y0U CAN USE THE G0SUB           WITH THE INPUT STATEMENT, WHEN
Y0U HAVE 'ANSWER YES 0R N0' AT SEVERAL PARTS 0F Y0UR PR0GRAM.
F0R AN EXAMPLE 0F THIS, SEE 'MIXER'.



*********************************************************************
```

```
USED      2.33 UNITS.
```

```
LISTNH

10 GØSUB 900
15 GØSUB 950
20 PRINT "HELLØ THERE, YØU SCHØLARS ØF THE DIGITAL CØMPUTER AND ØF ALL"
30 PRINT "ITS WØNDERS. THIS PRØGRAM IS TØ SHØW ØNE ØF THEM, NAMELY,"
40 PRINT "THE GØSUB CØMMAND. IT'S REALLY GREAT. "
50 GØSUB 950
60 GØSUB 900
70 GØSUB 950
80 PRINT "NØTICE HØW WE CAN USE THE GØSUB CØMMAND TØ DØ THE SAME THING"
90 PRINT "AT DIFFERENT PARTS ØF THE SAME PRØGRAM. "
100 GØSUB 950
110 PRINT "YØU CØULD MAKE UP A REALLY NIFTY BANK STATEMENT BY USING THE"
120 PRINT "GØSUB CØMMAND TØ SPRUCE IT UP WITH SPACES AND ASTERISKS ALL"
130 PRINT "ØVER THE PLACE. "
140 GØSUB 950
150 PRINT "GØSUB 950 MAKES A TRIPLE SPACE. GØSUB 900 MAKES A RØW ØF *'S."
160 PRINT "YØU CAN USE THE GØSUB         WITH THE INPUT STATEMENT, WHEN"
170 PRINT "YØU HAVE 'ANSWER YES ØR NØ' AT SEVERAL PARTS ØF YØUR PRØGRAM."
180 PRINT "FØR AN EXAMPLE ØF THIS,SEE 'MIXER'."
190 GØSUB 950
200 GØSUB 900
250 GØTØ 999
900 PRINT"*************************************************************"
910 RETURN
950 FØR I = 1 TØ 3
960 PRINT
970 NEXT I
980 RETURN
999 END
```

HOW TO WRITE A COMPUTER PROGRAM IN BASIC

A computer program tells the computer what to do. You can write your program in two lines.

INSTRUCTIONS

1 - First line - Tell the computer what to do.

To add: To add 46 + 89 + 13, you type

```
10 PRINT 46 + 89 + 13
```

To subtract: to subtract 23-14, you type

```
10 PRINT 23-14
```

To multiply: to multiply 3 x 23, you type

```
10 PRINT 3 * 23
```

To divide: to divide 48 ÷ 3, you type

```
10 PRINT 48/3
```

2 - Second line: The second line says

```
20 END
```

This tells the computer "that's all".

3 - Return - To start a new line, press the "Return" button. DON'T FORGET!

4 - Run - After you have typed in your program, type RUN and press the "return button. The computer will give you your answer.

Example: Add 46
89
13

YOU TYPE

```
10 PRINT 46 + 89 + 13 (return)
20 END                (return)
RUN                   (return)
```

THE COMPUTER TYPES

```
ARITH    11:16    M    THU   09/19/68
   148
RAN   1.33   SEC
```

148 is your answer.

Inventory Control - Lists and Tables - Data Files

In our program BANK and CHECK, we kept track of only one quantity, the balance in an account, B. Suppose you were in charge of a stockroom or warehouse with thousands of different things in stock? Before the computer, it was almost impossible to keep track of so many things. About the only way to check your stores was to shut down and take inventory.

Today, computers are used to keep track of inventory. They even can be used to reorder automatically when stock gets low. This is called "inventory control." INV is a simple example of how a computer keeps track of the stock.

INV, a Program Which Keeps Stock

Suppose you were in charge of pencils in a warehouse for a company that sold office supplies wholesale. You have in stock the following items:

Table - Inventory of Pencils

Hardness, A(I)	1	2	$2\frac{1}{2}$	3	4	7
Number of Boxes in Stock, N(I)	20	24	100	36	24	6
Subscript, I	1	2	3	4	5	6

You want the computer to keep track of pencils whenever an order comes in. You could give a different name, such as A, B, C, D, E, F, G, etc. for each item. But, with a warehouse with thousands of different items, you'd soon run out of letters.

A much simpler idea is to use one letter N for all pencils, but to give a number to each type of pencil. Thus N(1) stands for the number of No. 1 pencils, N(3) stands for the No. $2\frac{1}{2}$ pencils, etc. (See the table). These are called subscripted variables. (The letter N is the variable name and the numbers 1, 2, 3, 4, 5, 6 are the subscripts).

The Program and How it Works

The LIST of INV is shown below a RUN. The RUN is very simple. The computer asks you to write down the hardness (1, 2, 2.5, 3, 4, or 7) that you want and the number of boxes of pencils. You type in the answer, and the computer tells you how much stock is left.

Lines 10 and 20 put in the hardness and number of boxes. Lines 30-37 are a FOR-TO loop which tells the computer to READ the DATA in line 900, which gives the stock for each hardness of pencil. These DATA are given in the table. Notice that on each value of I, there is a certain hardness and number of boxes.

Lines 37-50 are another loop which tells the computer to search through the list to find the right value of I for the hardness, X, that was entered in line 20. As soon as the computer finds I, it leaves the loop at line 40 and goes to line 60, where it figures out the new stock of pencils for that item. Line 75 is an order to print out the new stock of pencils.

```
RUN


WLLINV    15:51   M  SAT 03/15/69

HARDNESS, NØ. ØF BØXES? 2.5,50

# 2.5  PENCILS : WITH 50   BØXES SHIPPED, 50   BØXES IN STØCK.



USED       3.17 UNITS.
LIST


WLLINV    15:51   M  SAT 03/15/69

10 PRINT "HARDNESS, NØ. ØF BØXES";
20 INPUT X,Y
30 FØR I = 1 TØ 6
35 READ A(I), N(I)
37 NEXT I
38 FØR I = 1 TØ 6
40 IF X = A(I) THEN 60
50 NEXT I
60 LET N(I) = N(I) -Y
70 PRINT
75 PRINT "#"; X; "PENCILS : WITH";Y;"BØXES SHIPPED,";N(I);"BØXES IN STØCK
900 DATA 1,20, 2,24, 2.5,100, 3,36, 4,24, 7,6
999 END
```

FIL, a DATA Files Program

Lets try to improve INV. The way INV works now, we'd have to type a new DATA statement in line 900 every time an order comes in. For example, line 900 would read:

```
900  DATA  1,20,  2,24,  2.5,50, 3,36,  4,24,  7,6
```

after the 50 boxes of No. 2.5 pencils were shipped out. The computer would hardly be worth while if we had to type up a new inventory every time we made up a shipment.

The DATA files feature of BASIC (see BASIC Extensions, p. 15) lets the computer do all the accounting work for us. First, we create a DATA file, which is a program named FIL. In it we put our inventory. Notice that we can leave out the word DATA, and just put in our numbers, separated by commas (or spaces).

Next, we rewrite our program, with a few changes. They are:

In line 5 (which must come before the computer is given a command to do anything, such as PRINT, LET, READ. Such a command is sometimes called an executable statement) is given the list of FILES. In this case,we have only one program WLLFIL. The READ statement in line 35 tells which file (No. 1) to use. Line 100 clears out FILE No. 1 to make way for the new inventory. Lines 110-130 make up the new DATA FILE automatically. Note in the LIST of FIL that the new inventory is correct. There are now only 50 boxes of No. 2.5 pencils.

```
NEW
NEW FILE NAME--WLLFIL

READY

1000 1,20, 2,24, 2.5,100, 3,36, 4,24, 7,6
SAVE

READY


ØLD
ØLD FILE NAME--WLLFIL


READY.

LISTNH

1000  1  20    2  24    2.5   100   3  36    4  24    7  6


ØLD
ØLD FILE NAME--WLLINV


READY.

RUN



WLLINV    15:14   M  SAT 03/15/69

HARDNESS, NØ. ØF BØXES? 2.5,50

# 2.5  PENCILS : WITH 50   BØXES SHIPPED, 50   BØXES IN STØCK.



USED      5.17 UNITS.
LISTNH
```

```
OLD
OLD FILE NAME--WLLFIL

READY.

LISTNH

1000   1  20    2  24    2.5   50    3  36    4  24    7  6
```

Problems:

1. Rewrite BANK or BILL to keep track of more than one account between dates of statements. (Banks use account numbers)
2. INV will "ship" as many boxes as you order, even if they aren't in stock. Rewrite INV to ship only as many boxes as are in stock, if too many are ordered.
3. Write a program that will take an order for several different hardnesses of pencils. Have it print up a bill of lading (a statement to go with the shipment that tells what is being sent). Have the bill of lading give an inventory of all pencils. Use a DATA statement for your order.
4. Write a program that will take an order, as in problem 3. Have it keep stock by typing out an order to the factory, whenever the stock is cut down by a shipment. However, the minimum order is 25 boxes. Tell the computer not to type up an order until the order exceeds 25 boxes. You can make up the name and address of the manufacturer and of your company, the jobber.

KENNY, A Demonstration Program to Illustrate Arithmetic Operations

This program is a simple one to illustrate common arithmetic operations on the computer. Note that only six digits are printed of the infinite decimal 1.11111... and that $9^{10} = 3.48678 \times 10^9$ is too large for the computer to print in decimal form. It therefore chooses to print the results in exponential form. The run and a list of the program are given below.

```
KENNY     14:56   M   THU 01/02/69

X =  10   Y =  9
X*Y =  90
X/Y =  1.11111
Y/X =  .9
X+Y =  19
X-Y =  1
Y-X = -1
Y↑X =  3.48678E+09
X↑Y =  1000000000

USED     2.00 UNITS.
LISTNH

10 LET X = 10
20 LET Y = 9
25 PRINT "X = ";X;"Y = ";Y
30 PRINT "X*Y = ";X*Y
40 PRINT "X/Y = ";X/Y
50 PRINT "Y/X = ";Y/X
60 PRINT "X+Y = ";X+Y
70 PRINT "X-Y = ";X-Y
80 PRINT "Y-X = ";Y-X
90 PRINT "Y↑X = ";Y↑X
100 PRINT "X↑Y = ";X↑Y
110 END
```

This program was written by Carolyn Coleman, Wilbur Cross High School.

```
L.C.M.     11:24     M     FRI     10/11/68

THIS PRØGRAM FINDS BØTH THE GREATEST CØMMØN DIVISØR AND
THE LØWEST CØMMØN MULTIPLE ØF TWØ NUMBERS, A AND B.

AFTER YØU ANSWER EACH QUESTIØN, BE SURE TØ PRESS THE 'RETURN'KEY
WHAT ARE YØUR VALUES FOR A, B [PUT A CØMMA BETWEEN THEM]
? 75,30

GREATEST CØMMØN FACTØR = 15
LØWEST CØMMØN MULTIPLE = 150

 DØ YØU WANT TØ RUN AGAIN [YES ØR NØ]? YES
WHAT ARE YØUR VALUES FØR A,B [PUT A CØMMA BETWEEN THEM]
? 30,75

GREATEST CØMMØN FACTØR = 15
LØWEST CØMMØN MULTIPLE = 150

 DØ YØU WANT TØ RUN AGAIN [YES ØR NØ]? YES
WHAT ARE YØUR VALUES FØR A,B [PUT A CØMMA BETWEEN THEM]
? 130,169

GREATEST CØMMØN FACTØR = 13
LØWEST CØMMØN MULTIPLE = 1690

 DØ YØU WANT TØ RUN AGAIN [YES ØR NØ]? I DØN'T KNØW
WØULD YØU PLEASE ANSWER 'YES' ØR 'NØ' ?
 DØ YØU WANT TØ RUN AGAIN [YES ØR NØ]? NØ
I'VE ENJØYED SERVING YØU. CØME AGAIN. NØW SAY 'GØØDBYE'

RAN     13.67 SEC
LISTNH

10  PRINT"THIS PRØGRAM FINDS BØTH THE GREATEST CØMMØN DIVISØR AND"
20  PRINT"THE LØWEST CØMMON MULTIPLE ØF TWØ NUMBERS, A AND B."
30  PRINT
40  PRINT"AFTER YØU ANSWER EACH QUESTIØN, BE SURE TØ PRESS THE'RETURN' KEY
45  PRINT "WHAT ARE YØUR VALUES FØR A,B (PUT A CØMMA BETWEEN THEM)"
50  INPUT A,B
58  LET Y = A*B
60  LET Q = INT(A/B)
70  LET R = A - Q*B
75  LET A =B
80  LET B = R
85  IF R >0 THEN 60
90  LET G=A
93  PRINT
95  PRINT "GREATEST CØMMØN FACTØR = ";G
100 LET M = Y/G
120 PRINT"LØWEST CØMMØN MULTIPLE = ";M
122 PRINT
125 PRINT"DØ YØU WANT TØ RUN AGAIN (YES ØR NØ)";
130 INPUT A$
140 IF A$ = "YES" THEN 45
150 IF A$ = "NØ" THEN 190
160 PRINT"WØULD YØU PLEASE ANSWER 'YES' ØR 'NØ' ?"
170 GØTØ 125
190 PRINT"I'VE ENJØYED SERVING YØU. CØME AGAIN. NØW SAY 'GØØDBYE'"
200 END
```

MIRR0R= A BASIC Program to Solve the Concave Mirror Equation

This program was used in an elementary physics course. The concave mirror satisfies the equation $I\ O = F^2$, where

I is the distance from image to the focal point, O is the distance from the object to focus, and F is the focal length of the mirror. The sign convention gives a positive value to I (or O), if the image (or object) is farther away from the mirror than the focus.

MIRR0R solves for any part of the equation, if the two other parts are given. In addition, the program distinguishes between real, virtual and infinitely distant objects or images. Finally, impossible cases are excluded.

Sample Run

The sample run is taken from an actual classroom demonstration where the students operated the teletype-writer.

First, the pupils used the program to process laboratory data. The values of I and O were placed in the computer, which calculated F. As larger values of O were used, the relative precision of the experiment increased, and the values for F converged to the value of .116 meter.

Then the program was turned around to solve for the image distance I, given O , the object distance and the focal length F. A relatively large value for the object distance (1 m) gave a small value for I (.013 m), which indicated that the image was close to the focus. An object at the focus (O=0) gave an image at infinity.

The Program

A list and flow chart are given for the program.

The program is simplified by the fact that image and object are inter-changeable. Thus the terms shown below

TABLE I - Interchangeable terms

image	↔	object
object	↔	image
I	↔	O
O	↔	I

```
RUN


MIRROR    11:15   M  MON 11/18/68

THIS IS A COMPUTER PROGRAM TO SOLVE THE EQUATION FOR A
SPHERICAL, CONCAVE MIRROR:   2
                            F  =  IO
WHERE F IS THE DISTANCE FROM THE MIRROR TO THE FOCUS. F IS
ALWAYS POSITIVE.
O IS THE DISTANCE FROM THE OBJECT TO THE FOCUS. O IS POS-
ITIVE ONLY IF THE FOCUS IS BETWEEN THE OBJECT AND THE MIRROR.
I IS THE DISTANCE FROM THE FOCUS TO THE IMAGE. IT IS POSI-
TIVE ONLY IF THE FOCUS IS BETWEEN THE IMAGE AND THE MIRROR.
     GIVE ME THE VALUES FOR ANY TWO OF THE QUANTITIES
O, I, OR F. I'LL SOLVE FOR THE THIRD. WHEN I ASK YOU,
TELL ME WHAT YOU WANT ME TO FIND. TYPE EITHER O, I OR F.
THEN PRESS THE RETURN KEY.

WHAT DO YOU WANT FROM ME: O, I OR F? F
TYPE THE VALUE FOR O, TYPE A COMMA, TYPE THE VALUE FOR I,
TYPE A COMMA, AND THEN TYPE THE UNITS. FOR EXAMPLE:5,6,CM.
DON'T FORGET THE UNITS.

WHAT ARE YOUR VALUES FOR I, O AND YOUR UNITS? 1.30,M←.008,M
F =  .10198  M

DO YOU WANT TO DO ANOTHER PROBLEM [ANSWER YES OR NO]? YES
WHAT DO YOU WANT FROM ME: O, I OR F? F
TYPE THE VALUE FOR O, TYPE A COMMA, TYPE THE VALUE FOR I,
TYPE A COMMA, AND THEN TYPE THE UNITS. FOR EXAMPLE:5,6,CM.
DON'T FORGET THE UNITS.

WHAT ARE YOUR VALUES FOR I, O AND YOUR UNITS? .94,.013,M
F =  .110544 M

DO YOU WANT TO DO ANOTHER PROBLEM [ANSWER YES OR NO]? YES
WHAT DO YOU WANT FROM ME: O, I OR F? F
TYPE THE VALUE FOR O, TYPE A COMMA, TYPE THE VALUE FOR I,
TYPE A COMMA, AND THEN TYPE THE UNITS. FOR EXAMPLE:5,6,CM.
DON'T FORGET THE UNITS.

WHAT ARE YOUR VALUES FOR I, O AND YOUR UNITS? .544,.025,M
F =  .116619 M

DO YOU WANT TO DO ANOTHER PROBLEM [ANSWER YES OR NO]? YES
WHAT DO YOU WANT FROM ME: O, I OR F? F
TYPE THE VALUE FOR O, TYPE A COMMA, TYPE THE VALUE FOR I,
TYPE A COMMA, AND THEN TYPE THE UNITS. FOR EXAMPLE:5,6,CM.
DON'T FORGET THE UNITS.

WHAT ARE YOUR VALUES FOR I, O AND YOUR UNITS? .385,.035,M
F =  .116082 M
```

```
DO YOU WANT TO DO ANOTHER PROBLEM [ANSWER YES OR NO]? YES
WHAT DO YOU WANT FROM ME: O, I OR F? I

TYPE YOUR VALUE FOR O, TYPEA COMMA, TYPE YOUR VALUE
FOR F, TYPE A COMMA, AND THEN TYPE YOUR UNITS. FOR EX:
WHAT ARE YOUR VALUES FOR O , F, UNITS          ? 5,10, CM 'RETURN'
 DON'T FORGET YOUR UNITS.

WHAT ARE YOUR VALUES FOR O , F, UNITS          ? 1,0,.116,M

INPUT DATA NOT IN CORRECT FORMAT. RETYPE IT.
? 1.0,.116,M
I =  .013456  M

DO YOU WANT TO DO ANOTHER PROBLEM [ANSWER YES OR NO]? YES
WHAT DO YOU WANT FROM ME: O, I OR F? I

TYPE YOUR VALUE FOR O, TYPEA COMMA, TYPE YOUR VALUE
FOR F, TYPE A COMMA, AND THEN TYPE YOUR UNITS. FOR EX:
WHAT ARE YOUR VALUES FOR O , F, UNITS          ? 5,10, CM 'RETURN'
 DON'T FORGET YOUR UNITS.

WHAT ARE YOUR VALUES FOR O , F, UNITS          ? 0,.116,M
IF THE OBJECT IS AT THE FOCUS, THEN THE IMAGE
IS AT INFINITY.
I IS INFINITE.

DO YOU WANT TO DO ANOTHER PROBLEM [ANSWER YES OR NO]? NO

I'VE ENJOYED SERVING YOU. COME AGAIN. NOW SAY 'GOODBYE' TO ME.



RAN     32.67 SEC

GOODBYE

*** OFF AT 11:40   ELAPSED TERMINAL TIME =      27 MIN.
```

```
LISTNH

10 PRINT "THIS IS A CØMPUTER PRØGRAM TØ SØLVE THE EQUATIØN FØR A "
20 PRINT "SPHERICAL, CØNCAVE MIRRØR:   2"
30 PRINT "                            F  =  IØ"
40 PRINT "WHERE F IS THE DISTANCE FRØM THE MIRRØR TØ THE FØCUS. F IS"
50 PRINT "ALWAYS PØSITIVE."
60 PRINT "Ø IS THE DISTANCE FRØM THE ØBJECT TØ THE FØCUS. Ø IS PØS-"
70 PRINT "ITIVE ØNLY IF THE FØCUS IS BETWEEN THE ØBJECT AND THE MIRRØR."
80 PRINT "I IS THE DISTANCE FRØM THE FØCUS TØ THE IMAGE. IT IS PØSI-"
90 PRINT "TIVE ØNLY IF THE FØCUS IS BETWEEN THE IMAGE AND THE MIRRØR."
100 PRINT "     GIVE ME THE VALUES FØR ANY TWØ ØF THE QUANTITIES"
110 PRINT "Ø, I, ØR F. I'LL SØLVE FØR THE THIRD. WHEN I ASK YØU,"
111 PRINT "TELL ME WHAT YØU WANT ME TØ FIND. TYPE EITHER Ø, I ØR F."
112 PRINT "THEN PRESS THE RETURN KEY."
113 PRINT
115 PRINT "WHAT DØ YØU WANT FRØM ME: Ø, I ØR F";
120 INPUT A$
130 IF A$ = "Ø" THEN 202
140 IF A$ = "I" THEN 260
150 IF A$ = "F" THEN 320
160 PRINT "SØRRY, I'M A STUPID CØMPUTER AND CAN'T UNDERSTAND YØU UNLESS"
165 PRINT "YØU TYPE EXACTLY WHAT YØU ARE SUPPØSED TØ. PLEASE RETYPE IT."
170 GØTØ 115
200 REM SØLVE FØR Ø, GIVEN F AND I.
202 LET B$ = "I"
210 LET C$ = "F"
220 LET D$ = "IMAGE"
230 LET E$ = "ØBJECT"
235 LET F$ = "Ø"
240 GØTØ 430
259 REM SØLVE FØR I, GIVEN F AND Ø.
260 LET B$ = "Ø"
270 LET C$ = "F"
280 LET D$ = "ØBJECT"
290 LET E$ = "IMAGE"
295 LET F$ = "I"
300 GØTØ 430
319 REM SØLVE FØR F, GIVEN I AND Ø.
320 PRINT "TYPE THE VALUE FØR Ø, TYPE A CØMMA, TYPE THE VALUE FØR I,"
325 PRINT "TYPE A CØMMA, AND THEN TYPE THE UNITS. FØR EXAMPLE:5,6,CM."
330 PRINT "DØN'T FØRGET THE UNITS."
333 PRINT
340 PRINT "WHAT ARE YØUR VALUES FØR I, Ø AND YØUR UNITS";
350 INPUT X,Y, G$
360 IF X*Y> 0 THEN 400
370 PRINT "IMPØSSIBLE. NEITHER Ø NØR I CAN BE ZERØ. ALSØ, BØTH MUST"
380 PRINT "HAVE THE SAME SIGN."
```

```
390 GØTØ 600
400 LET F = SQR (X*Y)
410 PRINT "F = "; F ; G$
420 GØTØ 600
430 PRINT
435 REM THIS IS A SUBRØUTINE TØ SØLVE FØR I IN TERMS ØF Ø AND F, ØR
436 REM VICE CERSA.
440 PRINT "TYPE YØUR VALUE FØR "; B$ ;", TYPEA CØMMA, TYPE YØUR VALUE"
450 PRINT "FØR "; C$ ;", TYPE A CØMMA, AND THEN TYPE YØUR UNITS. FØR EX:"
460 PRINT "WHAT ARE YØUR VALUES FØR "; B$ ;" , "; C$ ; ", UNITS",
465 PRINT "? 5,10, CM 'RETURN'"
466 PRINT " DØN'T FØRGET YØUR UNITS."
467 PRINT
470 PRINT "WHAT ARE YØUR VALUES FØR "; B$ ; " , " ; C$ ; ", UNITS",
480 INPUT X,Y, G$
485 IF Y>0 THEN 490
486 PRINT "IMPØSSIBLE. F CAN'T BE ZERØ ØR NEGATIVE."
487 GØTØ 600
490 IF X>0 THEN 530
500 IF X<0 THEN 550
510 PRINT "IF THE "; D$ ; " IS AT THE FØCUS, THEN THE " ; E$
515 PRINT "IS AT INFINITY."
520 PRINT F$; " IS INFINITE."
525 GØTØ 600
530 PRINT F$ ; " = "; Y↑2/X; " "; G$
540 GØTØ 600
550 IF ABS (X)<Y THEN 580
555 IF ABS(X) = Y THEN 590
560 PRINT "ØNLY THE "; E$ ; " IS REAL. THE "; D$ ; " IS VIRTUAL."
570 GØTØ 530
580 PRINT "ØNLY THE "; D$ ; " IS REAL. THE "; E$ ; " IS VIRTUAL."
585 GØTØ 530
590 PRINT "BØTH IMAGE AND ØBJECT ARE AT THE MIRRØR. I = Ø = -F = -";Y;G$
600 PRINT
605 PRINT "DØ YØU WANT TØ DØ ANØTHER PRØBLEM (ANSWER YES ØR NØ)";
610 INPUT G$
620 IF G$ = "YES" THEN 115
630 IF G$ = "NØ"  THEN 660
640 PRINT "SØRRY, I DIDN'T UNDERSTAND YØU."
650 GØTØ 605
660 PRINT
670 PRINT"I'VE ENJØYED SERVING YØU. CØME AGAIN. NØW SAY 'GØØDBYE' TØ ME."
999 END
```

can be interchanged, depending on whether F and I or F and 0 are given.

First, the operator tells the computer whether he wants it to solve for I, 0, or F (line 115). If F is wanted, the computer calls for 0 and I (lines 320-350). Unless the product of 0 and I are negative, which is impossible for a concave mirror, (line 360), the computer solves for F (lines 400-410).

If 0 or I are unknowns, the program follows the same subroutine in lines 430-600. Lines 202-240 or 260-300 merely insert the correct side of Table I in the statements. Line 486 excludes a negative focal length (convex mirror). Lines 490-510 present a triple branch. If the object (or image) is placed at the focus, the image (or object) is at infinity (lines 510-520).

If the object (or image) distance is negative (line 500 and lines 550-555), two cases need to be distinguished. If 0 (or I) is less in magnitude than F, the object (or image) is between the focus and the mirror; the object (or image) is real. Its mate is virtual. If 0 (or I) is greater in magnitude than F, the object (or image) is virtual, and lies beyond the mirror. Its mate is now real.

If the object (or image) distance is positive, both object and image are real (line 490). In any finite case, the computer prints out the solution for the unknown image (or object) distance (line 530).

It would be a relatively simple matter to modify this program to handle the cases of the convex mirror, converging or diverging lenses. All use the same equation. The only point that should be watched carefully is the sign convention.

MIXER - A Demonstration Program for Chemistry

Mixer is a useful program which demonstrates several important points in making up solutions of reagents:

1 - Stock solution of known concentration:
In the case of a standard solution, such as 0.1 N NaOH, MIXER tells the user how much is needed to make up a solution of any strength;

2 - Anhydrous reagent in solid form taken from list of reagents:
If the solid reagent is on the list of reagents, MIXER already has stored the molecular weight. It does all the calculations, given the volume and concentration of the desired solution. It also has stored the concentration of a saturated solution (at room temperature). If the program user tries to exceed this limit, he gets an error message which gives the molarity of the saturated solution. In the case of hydrates, a warning is sent and the correct amount of the common form of the hydrate is given.

3 - Common reagents in concentrated solutions:
Reagents such as ammonia, sulfuric acid, and hydrochloric acid, which commonly come in liquid form, are handled by the program. The correct amount of the concentrated liquid reagent is given. If the user tries to get a higher concentration than the commercial reagent, an error message tells him this is impossible. It also gives the molality of the concentrated reagent.

4 - Unlisted reagent. If a solid reagent is not listed, the user needs to know the molecular weight. If he gives this to the computer, MIXER tells him how much to use. There is no protection against trying to mix supersaturated solutions.

Examples of all four types of calculations are shown in the sample run.

The Program

The list of the program is given. By careful use of subscripted variables and subroutines, the program length is kept (barely) within the maximum permissable value. Instructions for entering additional data are given in lines 1400-1415. If necessary, these lines can be omitted to save space.

A unique feature of this program is the iteration variable X (see lines 35,121,202). This variable starts at 0 at the beginning of the run. It is set equal to 1, after the list of reagents is printed. This avoids the time-wasting possibility of printing the list twice during a run. To save time during successive runs, the user can enter a line to avoid printing the list of reagents. Just print:

```
35   LET  X = 1
RUN
```

```
RUN


MIXER     14:08   M   WED 02/26/69

DO YOU WANT TO MIX A SOLUTION? I'LL DO YOUR ARITHMETIC FOR
YOU.     WHENEVER I ASK YOU A QUESTION, TYPE IN THE ANSWER
RIGHT AFTER THE ? THEN PUSH THE RETURN BUTTON.

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? YES
WHEN YOU ANSWER THIS QUESTION, DON'T USE MORE THAN 15
SPACES TO DO IT. OTHERWISE I WON'T UNDERSTAND YOU.
WHAT IS THE FORMULA OF YOUR REAGENT? HCL
WHAT IS THE CONCENTRATION (IN MOLES/LITER) OF YOUR STOCK
SOLUTION? 1
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? .15
WHAT VOLUME (IN ML.) DO YOU WANT? 150

TO MAKE  150  ML. OF .15  MOLAR SOLUTION OF HCL, USE
 22.5 MILLILITERS OF STOCK SOLUTION OF
CONCENTRATION 1 M/L.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? YES

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? NO

STANDARD REAGENTS

              INORGANIC

NH3                 AMMONIA
CUSO4               COPPER SULFATE
HCL                 HYDROCHLORIC ACID
MGSO4               MAGNESIUM SULFATE
HNO3                NITRIC ACID
H3PO4               PHOSPHORIC ACID
KOH                 POTASSIUM HYDROXIDE
AGCL                SILVER CHLORIDE
AGNO3               SILVER NITRATE
NA2CO3              SODIUM CARBONATE
NAHCO3              SODIUM BICARBONATE
NACL                SODIUM CHLORIDE
NAOH                SODIUM HYDROXIDE
NA2S2O3             SODIUM THIOSULFATE
H2SO4               SULFURIC ACID

              ORGANIC

CH3COOH             ACETIC ACID
C6O7H8              CITRIC ACID
C6H12O6             GLUCOSE (DEXTROSE)
C12H22O11           SUCROSE (TABLE SUGAR)
```

```
IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)? YES
TYPE THE FORMULA OF YOUR REAGENT, EXACTLY AS IT IS IN THE LEFT
HAND COLUMN OF MY LIST. WHAT IS THE FORMULA OF YOUR REAGENT? NACL
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? 10
WHAT VOLUME (IN ML.) DO YOU WANT? 100

IMPOSSIBLE. THE HIGHEST POSSIBLE CONCENTRATION IS
 5.3  M/L. THIS IS FOR A SATURATED SOLUTION OF NACL.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? YES

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? NO

IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)? YES
TYPE THE FORMULA OF YOUR REAGENT, EXACTLY AS IT IS IN THE LEFT
HAND COLUMN OF MY LIST. WHAT IS THE FORMULA OF YOUR REAGENT? NACL
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? 1
WHAT VOLUME (IN ML.) DO YOU WANT? 100

TO MAKE  100  ML. OF 1 MOLAR SOLUTION OF NACL, USE
 5.84 GRAMS OF NACL.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? YES

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? NO

IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)? YES
TYPE THE FORMULA OF YOUR REAGENT, EXACTLY AS IT IS IN THE LEFT
HAND COLUMN OF MY LIST. WHAT IS THE FORMULA OF YOUR REAGENT? CUSO4
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? 1
WHAT VOLUME (IN ML.) DO YOU WANT? 1000

TO MAKE  1000 ML. OF 1 MOLAR SOLUTION OF CUSO4, USE
 159.6   GRAMS OF CUSO4.
YOUR REAGENT OFTEN COMES AS A HYDRATE. IN THAT CASE, USE
 249.68  GRAMS OF HYDRATE.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? YES

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? NO

IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)? N←YES
TYPE THE FORMULA OF YOUR REAGENT, EXACTLY AS IT IS IN THE LEFT
HAND COLUMN OF MY LIST. WHAT IS THE FORMULA OF YOUR REAGENT? HCL
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? 1
WHAT VOLUME (IN ML.) DO YOU WANT? 500

TO MAKE  500  ML. OF 1 MOLAR SOLUTION OF HCL, USE
 41.75   ML. OF CONC.( 37   PER CENT) HCL.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? YES

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? NO
```

```
IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)? YES
TYPE THE FORMULA OF YOUR REAGENT, EXACTLY AS IT IS IN THE LEFT
HAND COLUMN OF MY LIST. WHAT IS THE FORMULA OF YOUR REAGENT? H2SO4
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? 20
WHAT VOLUME (IN ML.) DO YOU WANT? 500

IMPOSSIBLE. THE HIGHEST POSSIBLE CONCENTRATION IS
 18.2073 M/L. THIS IS FOR A CONCENTRATED SOLUTION OF H2SO4.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? YES

IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION
(YES OR NO)? NO

IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)? NO
WHEN YOU ANSWER THIS QUESTION, DON'T USE MORE THAN 15
SPACES TO DO IT. OTHERWISE I WON'T UNDERSTAND YOU.
WHAT IS THE FORMULA OF YOUR REAGENT? PB(C2H3O2)3*3/2H20

INPUT DATA NOT IN CORRECT FORMAT. RETYPE IT.
? LEAD ACETATE

PLEASE GIVE THE MOLECULAR WEIGHT OF YOUR REAGENT, IN DECIMAL
FORM. FOR EXAMPLE, THE M.W. OF NAOH IS 40.0  IF YOU ARE
USING A HYDRATE, TYPE ITS M.W., NOT THAT OF THE ANHYDROUS
REAGENT.  WHAT IS THE MOLECULAR WEIGHT OF YOUR REAGENT? 343.07
WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT? .01
WHAT VOLUME (IN ML.) DO YOU WANT? 1000

TO MAKE  1000 ML. OF .01  MOLAR SOLUTION OF LEAD ACETATE, USE
 3.43 GRAMS OF LEAD ACETATE.

DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)? NO
IT'S BEEN MOLE FUN SOLVENT YOUR WEIGHTY PROBLEMS. I'M
SIMPLY REAGENT FURIOUS THAT WE HAVE TO DISSOLVE OUR
BONDS. BASE SEEING YOU. NOW YOU SAY GOODBYE TO ME.
0


USED     75.33 UNITS.
```

```
LISTNH

5 DIM F$(25),N$(25),M$(25),B$(25),W(25),D(25),M(25)
10 PRINT "DO YOU WANT TO MIX A SOLUTION? I'LL DO YOUR ARITHMETIC FOR"
20 PRINT "YOU.     WHENEVER I ASK YOU A QUESTION, TYPE IN THE ANSWER"
30 PRINT "RIGHT AFTER THE ? THEN PUSH THE RETURN BUTTON."
35 LET X = 0
40 PRINT
50 PRINT "IS YOUR REAGENT IN A STOCK SOLUTION OF KNOWN CONCENTRATION"
55 PRINT "(YES OR NO)";
60 INPUT S$
70 IF S$ = "YES" THEN 350
80 IF S$ = "NO" THEN 120
90 GOSUB 1000
100 GOTO 50
120 PRINT
121 IF X = 1 THEN 210
122 PRINT "STANDARD REAGENTS"
124 PRINT
126 PRINT TAB(10); "INORGANIC"
128 PRINT
130 READ N
135 FOR I = 1 TO N
140 READ F$(I),N$(I),M$(I),B$(I),W(I),D(I)
150 IF B$(I) = "A" THEN 170
160 READ M(I)
170 IF I<>16 THEN 190
180 PRINT
182 PRINT TAB(11); "ORGANIC"
184 PRINT
190 PRINT F$(1), N$(I);" "; M$(I)
200 NEXT I
202 LET X = 1
205 PRINT
210 PRINT "IS YOUR REAGENT ON MY LIST (ANSWER YES OR NO)";
220 INPUT A$
225 IF A$ = "YES" THEN 500
230 IF A$ = "NO"  THEN 250
235 GOSUB 1000
240 GOTO 210
250 GOSUB 1300
270 PRINT
280 PRINT "PLEASE GIVE THE MOLECULAR WEIGHT OF YOUR REAGENT, IN DECIMAL"
290 PRINT "FORM. FOR EXAMPLE, THE M.W. OF NAOH IS 40.0  IF YOU ARE"
300 PRINT "USING A HYDRATE, TYPE ITS M.W., NOT THAT OF THE ANHYDROUS"
310 PRINT "REAGENT.  WHAT IS THE MOLECULAR WEIGHT OF YOUR REAGENT";
320 INPUT W
330 LET M = 0
340 GOSUB 1050
345 GOSUB 1220
347 GOTO 1140
350 GOSUB 1300
352 PRINT "WHAT IS THE CONCENTRATION (IN MOLES/LITER) OF YOUR STOCK"
355 PRINT "SOLUTION";
360 INPUT C1
370 LET M = C1
380 GOSUB 1050
400 GOTO 1240
```

```
1350 LET M = D(I)
1360 LET W = W(I)
1370 GOSUB 1050
1380 GOSUB 1220
1390 RETURN
1400 REM INSTRUCTIONS FOR ENTERING DATA:
1401 REM IN LINE 1490, ENTER TOTAL NUMBER OF COMPOUNDS ON LIST
1402 REM IN LINES 1500-1900 ENTER DATA, IN FOLLOWING ORDER:
1403 REM FS(I) = CHEMICAL FORMULA
1404 REM NS(I) = 1ST WORD OF CHEM. NAME
1405 REM MS(I) = 2ND WORD OF CHEM. NAME (TYPE "" IF NAME IS 1 WORD)
1406 REM BS(I) = S FOR A REAGENT IN SOLUTION
1407 REM BS(I) = H FOR A SOLID REAG. WHICH COMES IN HYDRATED FORM
1408 REM BS(I) = A FOR A SOLID REAG. WHICH COMES ANHYDROUS
1410 REM W(I) = M.W. OF ANHYDROUS COMPOUND
1411 REM D(I) = DENSITY OF SOLUTION, IF BS(I) = "S"; OTHERWISE,
1412 REM D(I) = MAX. SOLUBILITY (M/L) OF ANHYD. COMPOUND
1413 REM M(I) = M.W. OF HYDRATE IF BS(I) = H
1414 REM M(I) = PER CENT (BY WGT.) OF ANHYD. REAG. IN SOLUTION, IF BS
1415 REM NO ENTRY FOR M(I) IF BS(I) = A
1490 DATA 19
1500 DATA NH3,AMMONIA,"",S,17.03,.9,19
1510 DATA CUSO4,COPPER,SULFATE,H,159.6,1.23,249.68
1520 DATA HCL,HYDROCHLORIC,ACID,S,36.46,1.18,37
1530 DATA MGSO4,MAGNESIUM,SULFATE,H,120.37,3.2,246.48
1540 DATA HNO3,NITRIC,ACID,S,63.01,1.41,68
1550 DATA H3PO4,PHOSPHORIC,ACID,S,98,1.834,85.3
1560 DATA KOH,POTASSIUM,HYDROXIDE,A,56.11,12.9
1570 DATA AGCL,SILVER,CHLORIDE,A,143.32,.000001
1580 DATA AGNO3,SILVER,NITRATE,A,169.87,8.7
1590 DATA NA2CO3,SODIUM,CARBONATE,H, 105.99,1.9,286.14
1600 DATA NAHCO3,SODIUM,BICARBONATE,A,84,1.1
1610 DATA NACL,SODIUM,CHLORIDE,A,58.44,5.3
1620 DATA NAOH,SODIUM,HYDROXIDE,A,40,18
1630 DATA NA2S2O3,SODIUM,THIOSULFATE,H,158.11,2.1,248.18
1640 DATA H2SO4,SULFURIC,ACID,S,98.08,1.841,97
1650 DATA CH3COOH,ACETIC,ACID,S,60.05,1.0491,100
1660 DATA C6O7H8,CITRIC,ACID,H,192.12,3.7,210.15
1670 DATA C6H12O6,GLUCOSE,(DEXTROSE),A,180.16,3.3
1680 DATA C12H22O11,SUCROSE,(TABLE SUGAR),A,342.30,2.6
1990 PRINT "IT'S BEEN MOLE FUN SOLVENT YOUR WEIGHTY PROBLEMS. I'M"
1992 PRINT "SIMPLY REAGENT FURIOUS THAT WE HAVE TO DISSOLVE OUR"
1993 PRINT "BONDS. BASE SEEING YOU. NOW YOU SAY GOODBYE TO ME."
1999 END
```

```
500 PRINT"TYPE THE FORMULA OF YOUR REAGENT, EXACTLY AS IT IS IN THE LEFT'
510 PRINT "HAND COLUMN OF MY LIST. WHAT IS THE FORMULA OF YOUR REAGENT";
520 INPUT FS
530 FOR I = 1 TO N
540 IF FS = FS(I) THEN 600
550 NEXT I
560 GOSUB 1000
570 GOTO 500
600 IF BS(I) = "A" THEN 700
610 IF BS(I) =."H" THEN 800
620 LET M = C1 = D(I)*M(I)*10/W(I)
630 GOSUB 1050
640 PRINT INT(100*V*C/C1+.5)/100;"ML. OF CONC.(";M(I);"PER CENT) ";FS;"."
650 GOTO 1140
700 GOSUB 1350
710 GOTO 1140
800 GOSUB 1350
810 PRINT "YOUR REAGENT OFTEN COMES AS A HYDRATE. IN THAT CASE, USE"
820 PRINT INT(M(I)*V*C/10 +.5)/100; "GRAMS OF HYDRATE."
830 GOTO 1140
1000 PRINT "SORRY. I DON'T UNDERSTAND YOU. PLEASE TYPE MORE CAREFULLY."
1010 RETURN
1050 PRINT "WHAT CONCENTRATION (MOLES/LITER) DO YOU WANT";
1060 INPUT C
1070 PRINT "WHAT VOLUME (IN ML.) DO YOU WANT";
1075 INPUT V
1080 IF M = 0 THEN 1120
1090 IF C<=M THEN 1120
1100 PRINT
1101 IF SS = "NO" THEN 1104
1102 LET GS = "YOUR STOCK"
1103 GOTO 1108
1104 IF BS(I) = "S" THEN 1107
1105 LET GS = "A SATURATED"
1106 GOTO 1108
1107  LET GS = "A CONCENTRATED"
1108 PRINT "IMPOSSIBLE. THE HIGHEST POSSIBLE CONCENTRATION IS"
1109 PRINT M; "M/L. THIS IS FOR ";GS;" SOLUTION OF ";FS;"."
1110 GOTO 1140
1120 GOSUB 1200
1130 RETURN
1140 PRINT
1150 PRINT "DO YOU WANT TO MIX UP ANOTHER SOLUTION (YES OR NO)";
1160 INPUT AS
1170 IF AS = "YES" THEN 40
1180 IF AS = "NO" THEN 1990
1190 GOSUB 1000
1195 GOTO 1140
1200 PRINT
1201 PRINT "TO MAKE "; V; "ML. OF"; C; "MOLAR SOLUTION OF "; FS;", USE"
1210 RETURN
1220 PRINT INT(M*V*C/10+.5)/100; "GRAMS OF "; FS; "."
1230 RETURN
1240 PRINT INT(100*V*C/C1+.5)/100;"MILLILITERS OF STOCK SOLUTION OF"
1250 PRINT "CONCENTRATION"; C1; "M/L."
1260 GOTO 1140
1300 PRINT "WHEN YOU ANSWER THIS QUESTION, DON'T USE MORE THAN 15"
1310 PRINT "SPACES TO DO IT. OTHERWISE I WON'T UNDERSTAND YOU."
1320 PRINT "WHAT IS THE FORMULA OF YOUR REAGENT";
1330 INPUT FS
1340 RETURN
```

NIM, A Game Which Involves Binary Numbers.

NIM, a program written at Dartmouth College, illustrates the game of nim. The rationale behind this game is explained in "Basic Programming", by Kemeny and Kurtz. Briefly, one writes the number of sticks as a binary number, adds up all the numbers in one column, makes a move to make all the sums of the same parity (even ~~or odd~~), and wins. The attached run shows how the computer won.

```
NIM          17:20        07/01/69

THIS IS THE GAME ØF NIM.
DØ YØU WANT INSTRUCTIØNS? YES
THE GAME IS PLAYED WITH A NUMBER ØF PILES ØF ØBJECTS.
ANY NUMBER ØF ØBJECTS  ARE REMØVED FRØM ØNE PILE BY YØU AND
THE MACHINE ALTERNATELY.  YØU MUST SPECIFY WHETHER WINNING
IS DEFINED AS TAKING ØR NØT TAKING THE LAST ØBJECT, THE
NUMBER ØF PILES IN THE GAME, AND HØW MANY ØBJECTS ARE
ØRIGINALLY IN EACH PILE.  EACH PILE MAY CØNTAIN A
DIFFERENT NUMBER ØF ØBJECTS.
THE MACHINE WILL SHØW ITS MØVE BY LISTING EACH PILE AND THE
NUMBER ØF ØBJECTS REMAINING IN THE PILES AFTER  EACH ØF ITS
MØVES.

ENTER WIN ØPTIØN - 1 TØ TAKE LAST, 2 TØ AVØID LAST? 2
ENTER NUMBER ØF PILES? 3
ENTER PILE SIZES
 1 ? 15
 2 ? 14
 3 ? 13
PILE  SIZE
 1  3
 2  14
 3  13
YØUR MØVE - PILE,NUMBER TØ BE REMØVED? 2,12
PILE  SIZE
 1  3
 2  2
 3  1
YØUR MØVE - PILE,NUMBER TØ BE REMØVED? 3 DELETED
1,2
PILE  SIZE
 1  1
 2  1
 3  1
YØUR MØVE - PILE,NUMBER TØ BE REMØVED? 1,1
MACHINE WINS

USED     .49 UNITS
```

14 1110
13 1101

Sum 3322

Computer moves:

3 0011
14 1110
13 1101

Sum 2222 (all even)

Player moves:

3 0011
2 0010
13 1101

1122

Computer moves:

3 0011
2 0010
1 0001

0022 (all even)

Player moves:

1 0001
2 0010
1 0001

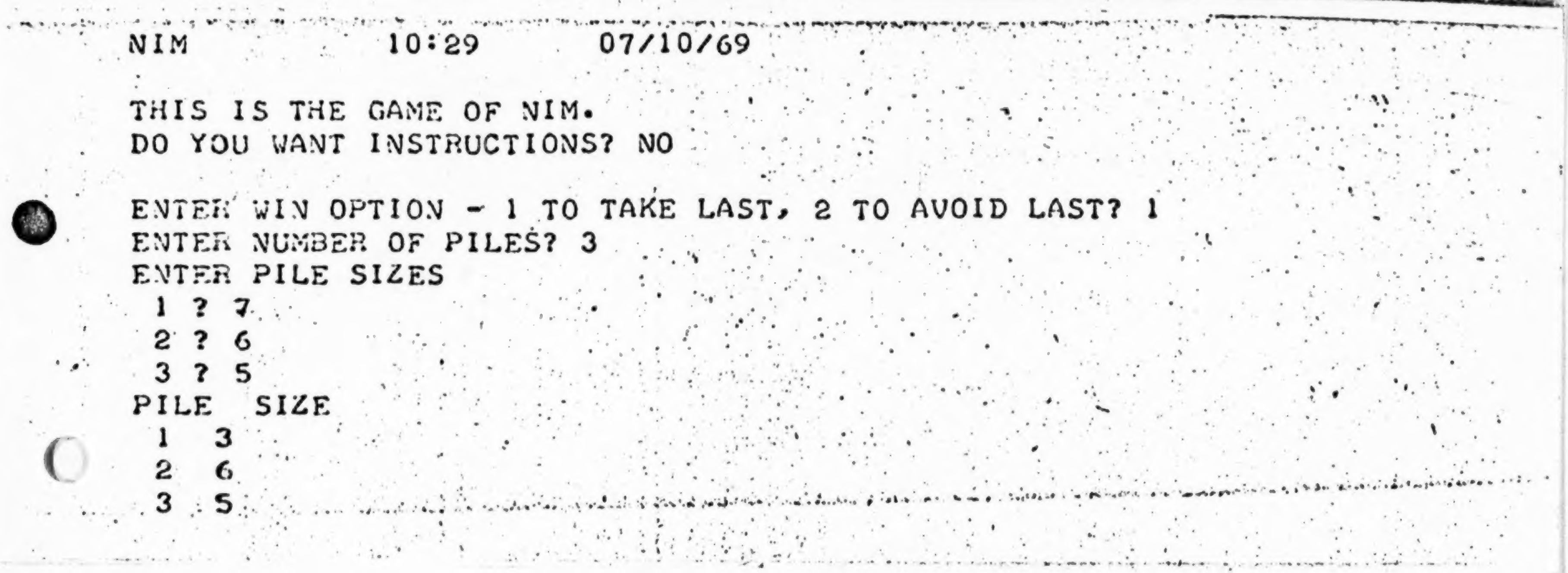
0012

Computer moves:

1 0001
1 0001
1 0001

0003, which is a certain win, as the player must take the last stick.

Problem: Show that the same strategy was used to win NIM, when the last person to remove a stick is the winner. (see below)



```
NIM          10:29      07/10/69

THIS IS THE GAME OF NIM.
DO YOU WANT INSTRUCTIONS? NO

ENTER WIN OPTION - 1 TO TAKE LAST, 2 TO AVOID LAST? 1
ENTER NUMBER OF PILES? 3
ENTER PILE SIZES
 1 ? 7
 2 ? 6
 3 ? 5
PILE  SIZE
 1   3
 2   6
 3   5
```

```
YOUR MOVE - PILE,NUMBER TO BE REMOVED? 1,3
PILE  SIZE
 1  0
 2  5
 3  5
YOUR MOVE - PILE,NUMBER TO BE REMOVED? 2,3
PILE  SIZE
 1  0
 2  2
 3  2
YOUR MOVE - PILE,NUMBER TO BE REMOVED? 2,1
PILE  SIZE
 1  0
 2  1
 3  1
YOUR MOVE - PILE,NUMBER TO BE REMOVED? 2,1
PILE  SIZE
 1  0
 2  0
 3  0
MACHINE WINS


USED      .52 UNITS
RUN

NIM            10:30       07/10/69

THIS IS THE GAME OF NIM.
DO YOU WANT INSTRUCTIONS? NO

ENTER WIN OPTION - 1 TO TAKE LAST, 2 TO AVOID LAST? 1
ENTER NUMBER OF PILES? 3
ENTER PILE SIZES
 1 ? 7
 2 ? 6
 3 ? 1
PILE  SIZE
 1  7
 2  0
 3  1
YOUR MOVE - PILE,NUMBER TO BE REMOVED? 1,6
PILE  SIZE
 1  1
 2  0
 3  0
YOUR MOVE - PILE,NUMBER TO BE REMOVED? 1,1
MACHINE LOSES


USED      .42 UNITS
LISTNH
100 REM  *  GAMNIM  *
110 DIM A(100),B(100,10),D(2)
120 PRINT "THIS IS THE GAME OF NIM."
130 PRINT "DO YOU WANT INSTRUCTIONS";
140 INPUT ZS
150 IF ZS="NO" THEN 260
151 IF ZS="YES" THEN 160
152 PRINT "INCORRECT ANSWER.  PLEASE TYPE 'YES' OR 'NO'";
153 GOTO 140
160 PRINT "THE GAME IS PLAYED WITH A NUMBER OF PILES OF OBJECTS."
170 PRINT "ANY NUMBER OF OBJECTS  ARE REMOVED FROM ONE PILE BY YOU AND"
180 PRINT "THE MACHINE ALTERNATELY.  YOU MUST SPECIFY WHETHER WINNING"
```

```
190 PRINT "IS DEFINED AS TAKING OR NOT TAKING THE LAST OBJECT, THE"
200 PRINT "NUMBER OF PILES IN THE GAME, AND HOW MANY OBJECTS ARE"
210 PRINT "ORIGINALLY IN EACH PILE.  EACH PILE MAY CONTAIN A"
220 PRINT "DIFFERENT NUMBER OF OBJECTS."
230 PRINT "THE MACHINE WILL SHOW ITS MOVE BY LISTING EACH PILE AND THE"
240 PRINT "NUMBER OF OBJECTS REMAINING IN THE PILES AFTER  EACH OF ITS"
250 PRINT "MOVES."
260 PRINT
270 PRINT "ENTER WIN OPTION - 1 TO TAKE LAST, 2 TO AVOID LAST";
280 INPUT W
290 IF W=1 THEN 310
300 IF W<>2 THEN 270
310 PRINT "ENTER NUMBER OF PILES";
320 INPUT N
330 IF N>100 THEN 310
340 IF N<1 THEN 310
350 IF N<>INT(N) THEN 310
360 PRINT "ENTER PILE SIZES"
```

```
370 FOR I=1 TO N
380 PRINT I;
390 INPUT A(I)
400 IF A(I)>2000 THEN 380
410 IF A(I)<1 THEN 380
420 IF A(I)<>INT(A(I)) THEN 380
430 NEXT I
440 IF W=1 THEN 680
450 LET C=0
460 FOR I=1 TO N
470 IF A(I)=0 THEN 510
480 LET C=C+1
490 IF C=3 THEN 580
500 LET D(C)=I
510 NEXT I
520 IF C=2 THEN 660
530 IF A(D(1))>1 THEN 560
540 PRINT "MACHINE LOSES"
550 STOP
560 PRINT "MACHINE WINS"
570 STOP
580 LET C=0
590 FOR I=1 TO N
600 IF A(I)>1 THEN 680
610 IF A(I)=0 THEN 630
620 LET C=C+1
630 NEXT I
640 IF C/2<>INT(C/2) THEN 540
650 GOTO 680
660 IF A(D(1))=1 THEN 560
670 IF A(D(2))=1 THEN 560
680 FOR I=1 TO N
690 LET E=A(I)
700 FOR J=0 TO 10
710 LET F=E/2
720 LET B(I,J)=2*(F-INT(F))
730 LET E=INT(F)
740 NEXT J
750 NEXT I
760 FOR J=10 TO 0 STEP -1
770 LET C=0
780 LET H=0
790 FOR I=1 TO N
800 IF B(I,J)=0 THEN 850
810 LET C=C+1
820 IF A(I)<=H THEN 850
830 LET H=A(I)
840 LET G=I
850 NEXT I
860 IF C/2<>INT(C/2) THEN 930
870 NEXT J
880 LET E=INT(N*RND(Z)+1)
890 IF A(E)=0 THEN 880
900 LET F=INT(A(E)*RND(Z)+1)
910 LET A(E)=A(E)-F
920 GO TO 1120
930 LET A(G)=0
940 FOR J=0 TO 10
950 LET B(G,J)=0
```

```
960 LET C=0
970 FOR I=1 TO N
980 IF B(I,J)=0 THEN 1000
990 LET C=C+1
1000 NEXT I
1010 LET A(G)=A(G)+2*(C/2-INT(C/2))*2↑J
1020 NEXT J
```

```
1030 IF W=1 THEN 1120
1040 LET C=0
1050 FOR I=1 TO N
1060 IF A(I)>1 THEN 1120
1070 IF A(I)=0 THEN 1090
1080 LET C=C+1
1090 NEXT I
1100 IF C/2<>INT(C/2) THEN 1120
 110 LET A(G)=1-A(G)
1120 PRINT "PILE  SIZE"
1130 FOR I=1 TO N
1140 PRINT I;A(I)
1150 NEXT I
1160 IF W=2 THEN 1190
1170 GOSUB 1310
1180 IF Z=1 THEN 560
1190 PRINT "YOUR MOVE - PILE,NUMBER TO BE REMOVED";
1200 INPUT X,Y
1210 IF X>N THEN 1190
1220 IF X<1 THEN 1190
1230 IF X<>INT(X) THEN 1190
1240 IF Y>A(X) THEN 1190
1250 IF Y<1 THEN 1190
1260 IF Y<>INT(Y) THEN 1190
1270 LET A(X)=A(X)-Y
1280 GOSUB 1310
1290 IF Z=1 THEN 540
1300 GO TO 440
1310 LET Z=0
1320 FOR I=1 TO N
1330 IF A(I)=0 THEN 1350
1340 RETURN
1350 NEXT I
 360 LET Z=1
1370 RETURN
1380 END
```

PI - Sample Programs to Show How Data is Fed into the Computer in BASIC

Whenever we solve a problem on the computer, we have to give it data of one kind or another. Usually the data are numbers. For example, suppose we want to solve the following problem:

Find -2π, -4π, -6π, -8π, -10π. To solve it, we have to put in six numbers: π, 2, 4, 6, 8 and 10.

A clumsy way to handle this problem is to write all the numbers in the program with LET statements, such as:

```
10 LET Z = -3.14159
20 LET Y = 2
30 PRINT Z*Y
40 LET Y = 4
50 PRINT Z*Y
60 LET Y = 6
70 PRINT Z*Y
80 LET Y = 8
90 PRINT Z*Y
100 LET Y = 10
110 PRINT Z*Y
999 END
```

This is clearly a waste of time and space. It is not using the great power of the computer to do one operation over and over again, with only a small change in the program. There are three ways to do this. These are shown in the next few pages. (See also the three examples in the back of the "Learner's Guide")

1 - The DATA statement.

The computer acts as a data processor if we write a program to read the data and to "process it." In the program, PI, on p. 2, line 10 tells the computer to read the first piece of data (2) which is in line 60. Lines 20 and 40 tell the computer to take this number (2) and multiply it by -π and print the result. (The DATA statement can go anywhere in the program.)

Since the computer is very stupid, it would do this for us and then go on to the next statement in the program, line 999 END. To keep it from stopping before its work is done, we tell it to go back and start all over again:

```
50 GØ TØ 10
```

It keeps doing this until it runs out of data. (The data are separated by commas. If you run out of space on a line, you can start more DATA statements with lines 61,62, etc.)

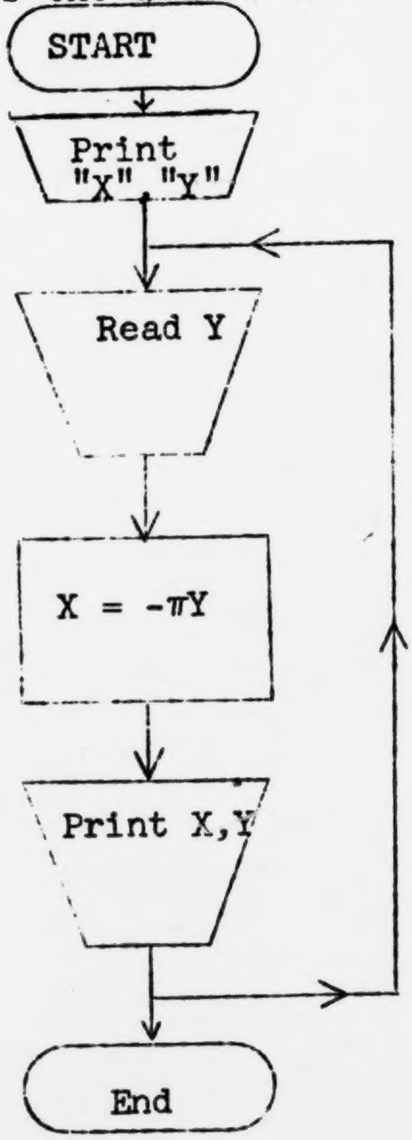
LOOPS - FLOW CHARTS

This program is an example of a loop, a very important part of almost any computer program. It is called a loop, because the computer goes around and does the same thing over and over again, until it is told to stop. In this case, the computer always stops when it runs out of data. (The statement ØUT ØF DATA IN 10 refers to line 10, where the READ statement is given).

A flow chart is a diagram which shows how the computer does its job. Flow charts are most useful if they are drawn up before you write your program. A flow chart for this program is shown in the figure.

A sample run for the program PI illustrates the use of the DATA statement.

```
NEW
NEW FILE NAME--PI
READY.

TAPE
READY.


5 PRINT "X","Y"
10 READ Y
20 LET X = -3.14159 *Y
40 PRINT X, Y
50 GØ TØ 10
60 DATA 2,4,6,8,10
999 END


KEY
READY.

RUN


PI      11:58     M TUE 12/03/68

X                Y
-6.28318         2
-12.5664         4
-18.8495         6
-25.1327         8
-31.4159         10

ØUT ØF DATA  IN 10


USED      1.67 UNITS.
```



Flow Chart for PI

Notice that the instruction 5 PRINT "X","Y" is outside of the loop. This is to print the heading of the table only once.

THE FØR... TØ STATEMENT

Whenever the numbers come in equal steps, the FØR.... TØ statement is an easy way to make up a loop. The list and run of PI2 shows how such a program runs. Y starts at 2, runs in steps of 2. The computer goes on to the next line in the program after Y has reached 10. (Steps can be plus or minus, whole numbers or decimals. Often the STEP is 1, in which case the STEP statement can be left out.) The statement in line 40 NEXT Y is really short hand for LET Y = Y + 2 and GØ TØ 20.

```
NEW
NEW FILE NAME--PI2
READY.

TAPE
READY.


5 PRINT "X","Y"
10 FOR Y = 2 TØ 10 STEP 2
20 LET X = - 3.14159*Y
30 PRINT X,Y
40 NEXT Y
999 END


KEY
READY.

RUN


PI2      12:01    M TUE 12/03/68

X            Y
-6.28318      2
-12.5664      4
-18.8495      6
-25.1327      8
-31.4159      10



USED    1.67 UNITS.
```

INPUT

The third way to put numbers in the computer is to use the INPUT statement. The computer prints a (?) and waits for you to type in an answer. Note that the (;) after the PRINT statement in line 20 close packs the answer after the question mark. A comma(,) makes the computer move over a wider space for an answer. With no punctuation, the INPUT statement puts the question mark on a new line.

Note that this program has no way of ending the loop. A STØP statement, use of the "break" key, or disconnecting the computer (CLR button) is needed to end the run.

Also notice that INPUT statements eat up a great deal of expensive computer time. If you have a program with many questions and answers, it may save money to use a faster computer, such as the G.E. Mark II. A sample run is shown in PI3.

```
NEW
NEW FILE NAME--PI3
READY.

TAPE
READY.


20 PRINT "FURNISH A VALUE ØF Y";
30 INPUT Y
33 LET X = - 3.14159*Y
35 PRINT "X","Y"
37 PRINT X,Y
40 GØ TØ 20
999 END

KEY
READY.

RUN

PI3     12:02   M TUE   12/03/68

FURNISH A VALUE ØF Y?  2
X              Y
-6.28318        2
FURNISH A VALUE ØF Y?  4
X              Y
-12.5664        4
FURNISH A VALUE ØF Y?  6
X              Y
-18.8495        6
FURNISH A VALUE ØF Y?  8
X              Y
-25.1327        8
FURNISH A VALUE ØF Y?  10
X              Y
-31.4159        10
FURNISH A VALUE ØF Y?  STØP
USED      7.67 UNITS.
```

COMPUTER CLASS ASSIGNMENT

1.- Use a data statement to put in X. Print a table of X, X^2, X^3 for each integer from 1 to 10. Make neat headings for your table.

2 - Use the FØR...TØ statement to do the same for X = 2, 5,8,11.

3 - Use the input statement to do the same for a few values of X. Print your results like this:

```
X = 2    X↑2 = 4   X↑3 = 8
```

Num, ØRD and A2Z - Filing with the Computer

These programs show how the computer can be used to take in data, sort it, and file it away in tables which are in order. In other words, the computer does the job of a clerk, and does it much faster.

These programs and their preparation also show various features of the BASIC language:

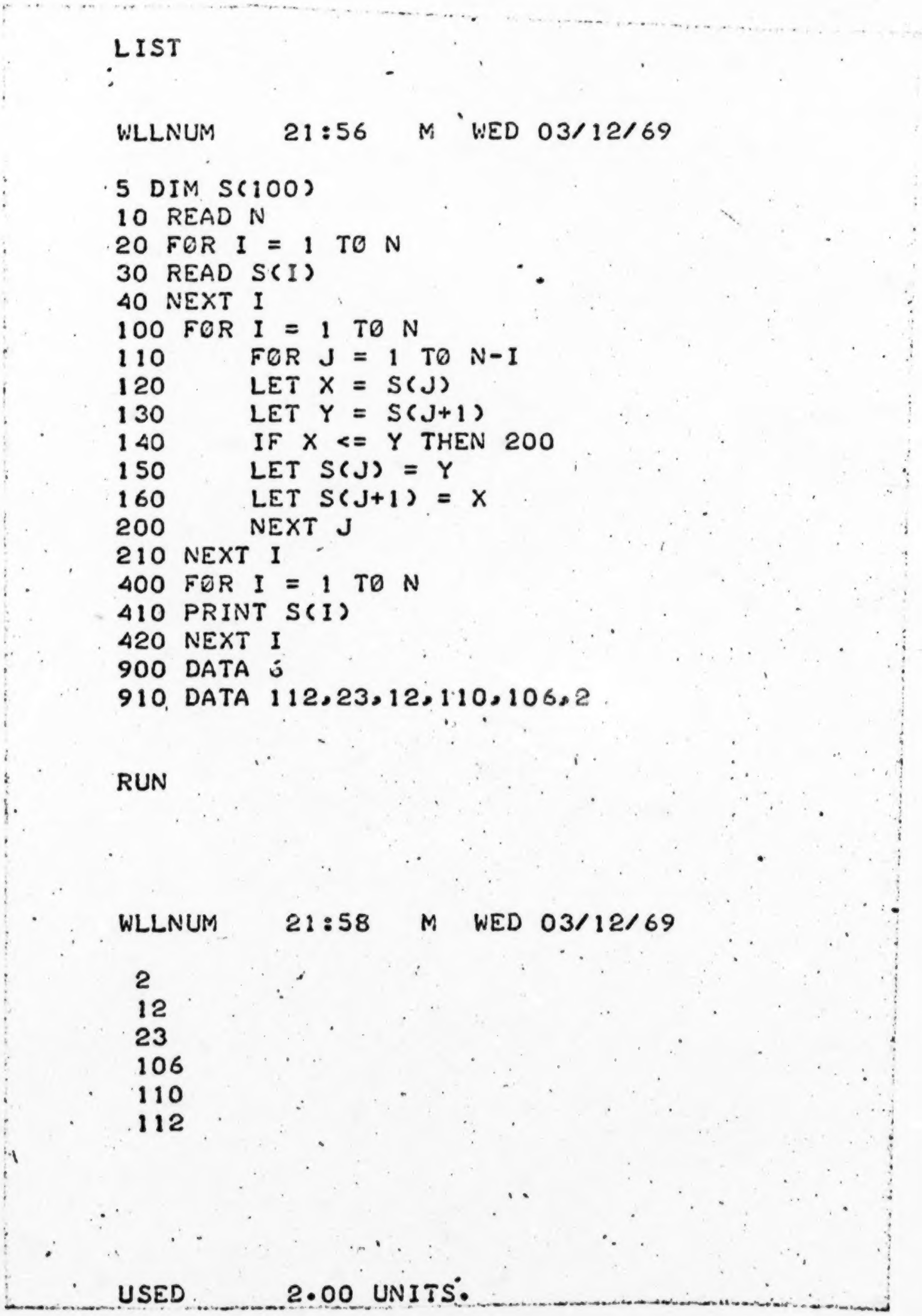
1. Lists and tables (subscripted variables).
2. The FØR - TØ statement.
3. EDIT Commands.

What the Programs Do

First, let's look at NUM, a program which takes a list of numbers and arranges them in order. Note that, to use this program, the DATA in line 900 give the number of entries (how many numbers you want to have sorted). In line 910, the DATA give the numbers to be sorted.

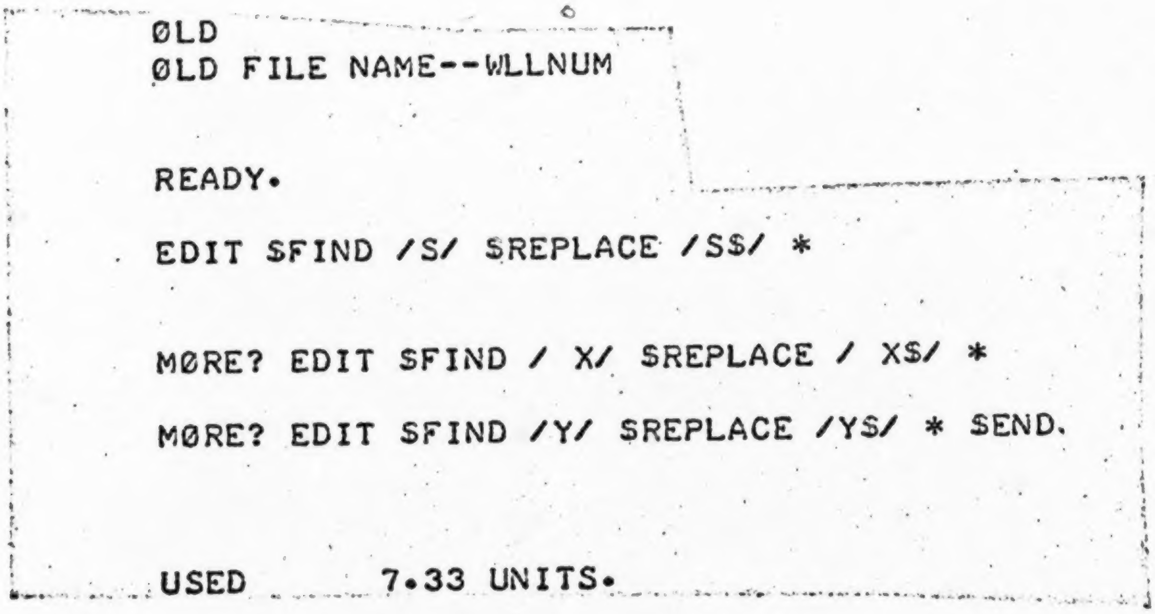
Second, note that a few small changes can make NUM, a program to sort numbers, into A2Z, a program to sort words and put them into alphabetical order. To do this, we merely change every variable name (S, X and Y) into a string variable name (S$ X$ and Y$). This could be done by retyping the program. A simple way to do it is to use EDIT commands.

First, we show, the list of NUM and a sample run:



```
LIST


WLLNUM    21:56   M  WED 03/12/69

5 DIM S(100)
10 READ N
20 FØR I = 1 TØ N
30 READ S(I)
40 NEXT I
100 FØR I = 1 TØ N
110     FØR J = 1 TØ N-I
120     LET X = S(J)
130     LET Y = S(J+1)
140     IF X <= Y THEN 200
150     LET S(J) = Y
160     LET S(J+1) = X
200     NEXT J
210 NEXT I
400 FØR I = 1 TØ N
410 PRINT S(I)
420 NEXT I
900 DATA 6
910 DATA 112,23,12,110,106,2


RUN



WLLNUM    21:58   M  WED 03/12/69

 2
 12
 23
 106
 110
 112



USED      2.00 UNITS.
```

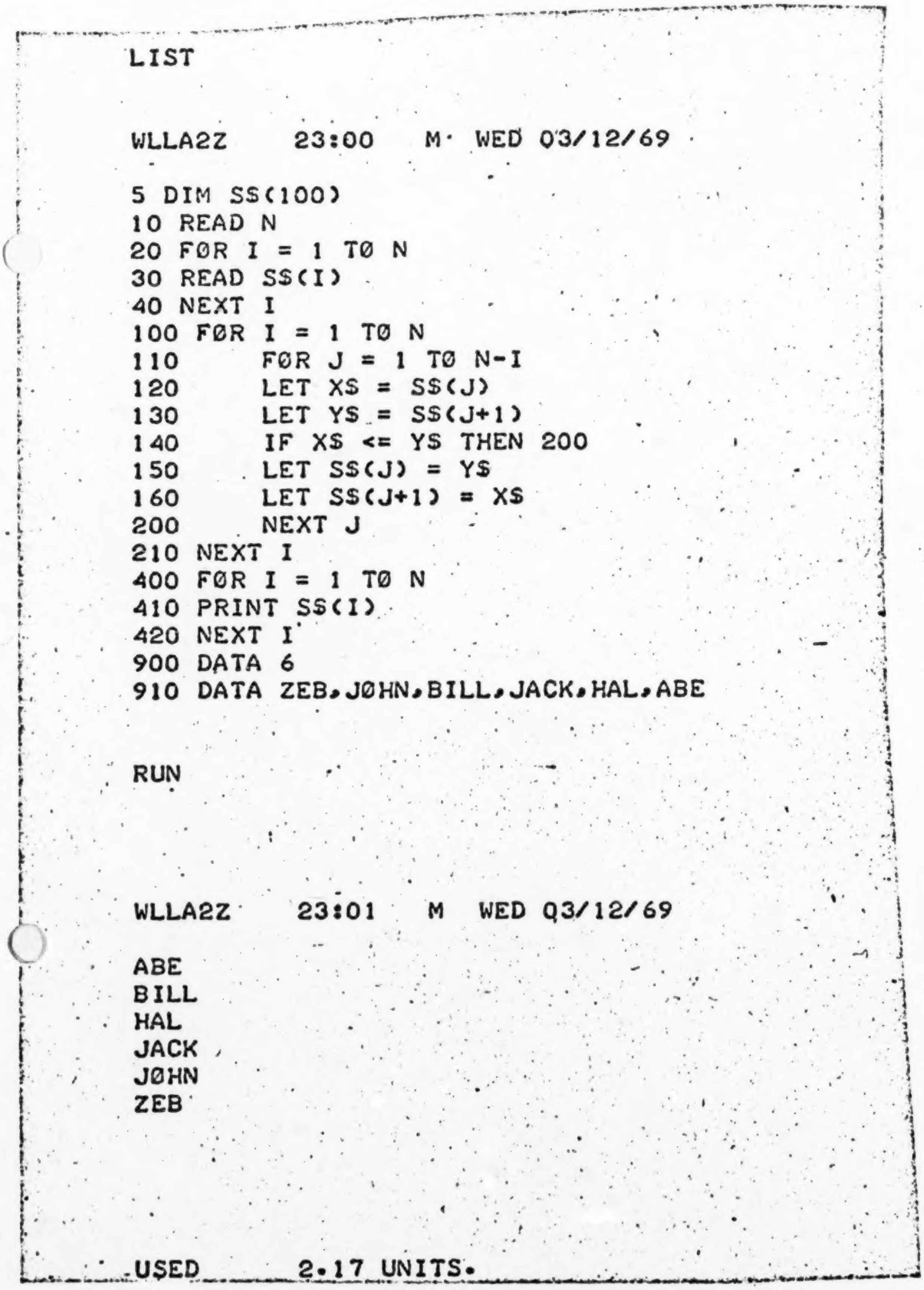
Next, we give it the correct EDIT commands (see "EDITING COMMANDS" Manual) to change the variable (S, X and Y) into string variables (note that the second command is $FIND / X/, not $FIND /X/. The reason for this is that X occurs in the word NEXT, but without a space)



```
ØLD
ØLD FILE NAME--WLLNUM


READY.

EDIT $FIND /S/ $REPLACE /S$/ *

MØRE? EDIT $FIND / X/ $REPLACE / X$/ *

MØRE? EDIT $FIND /Y/ $REPLACE /Y$/ * $END.


USED      7.33 UNITS.
```

We now change the DATA in line 910 to a list of words, rename the program A2Z, and we have a program to alphabetize word or name lists.

The LIST and a RUN of A2Z is shown below:

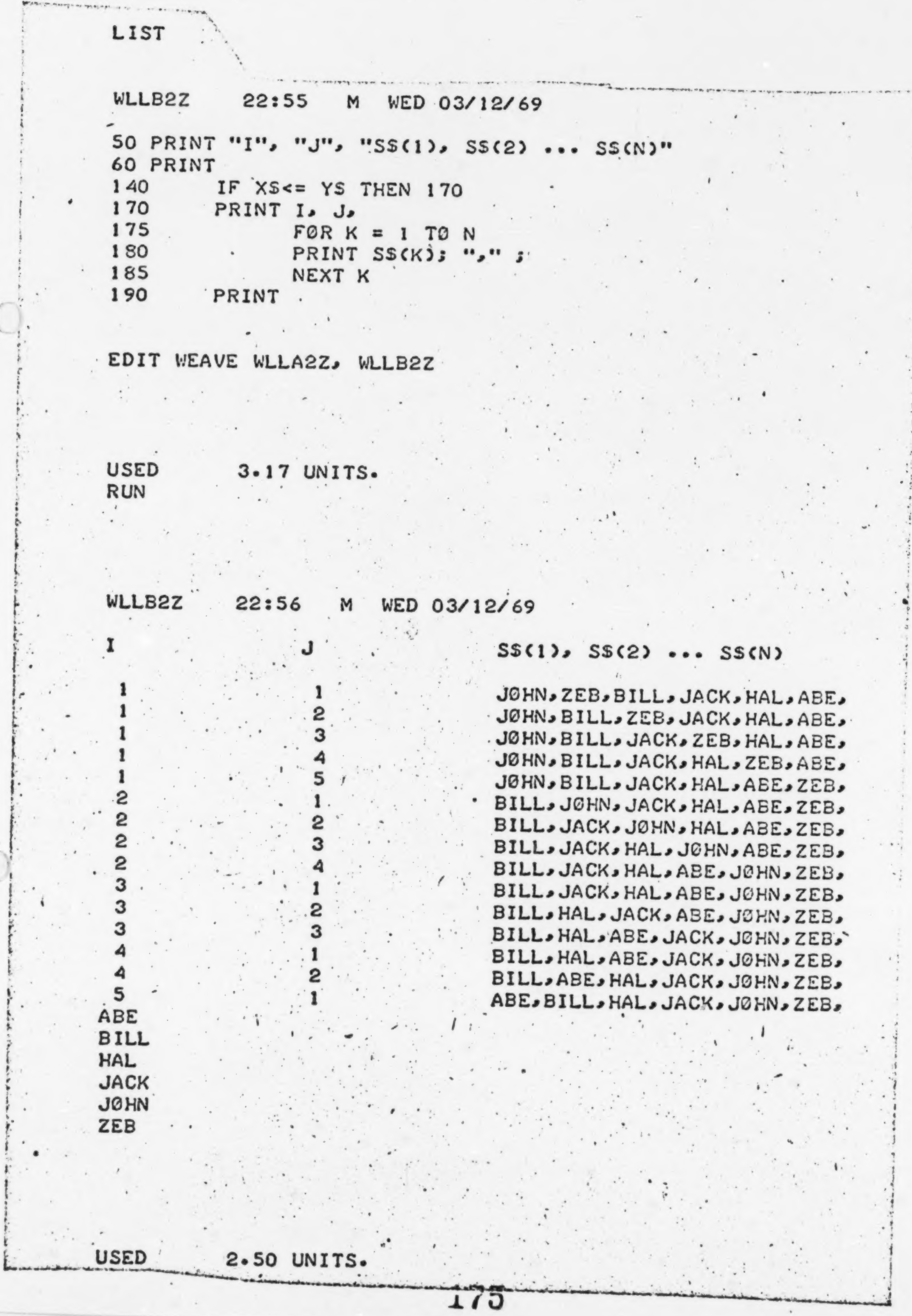


```
LIST


WLLA2Z     23:00    M   WED 03/12/69

5 DIM S$(100)
10 READ N
20 FØR I = 1 TØ N
30 READ S$(I)
40 NEXT I
100 FØR I = 1 TØ N
110      FØR J = 1 TØ N-I
120      LET X$ = S$(J)
130      LET Y$ = S$(J+1)
140      IF X$ <= Y$ THEN 200
150      LET S$(J) = Y$
160      LET S$(J+1) = X$
200      NEXT J
210 NEXT I
400 FØR I = 1 TØ N
410 PRINT S$(I)
420 NEXT I
900 DATA 6
910 DATA ZEB,JØHN,BILL,JACK,HAL,ABE


RUN




WLLA2Z     23:01    M   WED 03/12/69

ABE
BILL
HAL
JACK
JØHN
ZEB





USED      2.17 UNITS.
```

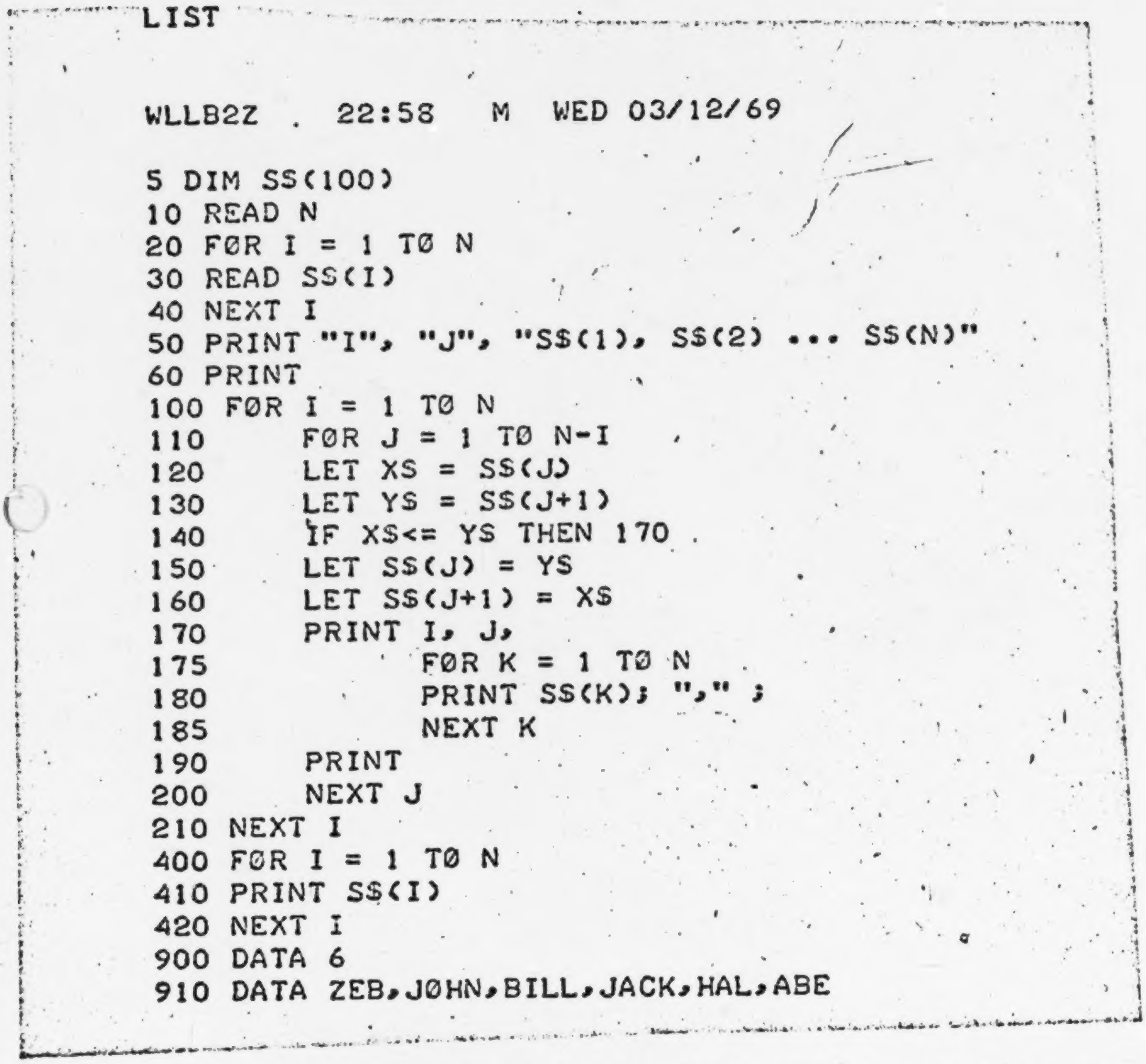
How the Programs Work - "Debugging"

Since sorting programs are so important in data processing, we'll ask the computer to tell us how it works. In"debugging" a program (finding the "bugs" in it, and getting them out), it is very useful to put in extra PRINT commands to tell the computer to let us know what it is doing in the middle of its job. We do this in A2Z, by putting in additional lines 50 - 60, 170-190 and by changing the THEN 200 to THEN 170 in line 140. We do this by use of the EDIT WEAVE Command, which is very useful for combining programs. (Note that there is a line 170 in both programs. The line 170 of the last program is saved.

Notice that the LIST of the new program orders the computer to PRINT the words in their new order every time the computer compares two words:



```
LIST

WLLB2Z     22:55    M   WED 03/12/69

50 PRINT "I", "J", "S$(1), S$(2) ... S$(N)"
60 PRINT
140        IF X$<= Y$ THEN 170
170        PRINT I, J,
175              F0R K = 1 T0 N
180              PRINT S$(K); "," ;
185              NEXT K
190        PRINT

EDIT WEAVE WLLA2Z, WLLB2Z



USED       3.17 UNITS.
RUN



WLLB2Z     22:56    M   WED 03/12/69

I              J              S$(1), S$(2) ... S$(N)

 1             1              J0HN,ZEB,BILL,JACK,HAL,ABE,
 1             2              J0HN,BILL,ZEB,JACK,HAL,ABE,
 1             3              J0HN,BILL,JACK,ZEB,HAL,ABE,
 1             4              J0HN,BILL,JACK,HAL,ZEB,ABE,
 1             5              J0HN,BILL,JACK,HAL,ABE,ZEB,
 2             1              BILL,J0HN,JACK,HAL,ABE,ZEB,
 2             2              BILL,JACK,J0HN,HAL,ABE,ZEB,
 2             3              BILL,JACK,HAL,J0HN,ABE,ZEB,
 2             4              BILL,JACK,HAL,ABE,J0HN,ZEB,
 3             1              BILL,JACK,HAL,ABE,J0HN,ZEB,
 3             2              BILL,HAL,JACK,ABE,J0HN,ZEB,
 3             3              BILL,HAL,ABE,JACK,J0HN,ZEB,
 4             1              BILL,HAL,ABE,JACK,J0HN,ZEB,
 4             2              BILL,ABE,HAL,JACK,J0HN,ZEB,
 5             1              ABE,BILL,HAL,JACK,J0HN,ZEB,
ABE
BILL
HAL
JACK
J0HN
ZEB



USED       2.50 UNITS.
```



```
LIST


WLLB2Z  .  22:58   M  WED 03/12/69

5 DIM S$(100)
10 READ N
20 FØR I = 1 TØ N
30 READ S$(I)
40 NEXT I
50 PRINT "I", "J", "S$(1), S$(2) ... S$(N)"
60 PRINT
100 FØR I = 1 TØ N
110       FØR J = 1 TØ N-I
120       LET X$ = S$(J)
130       LET Y$ = S$(J+1)
140       IF X$<= Y$ THEN 170
150       LET S$(J) = Y$
160       LET S$(J+1) = X$
170       PRINT I, J,
175             FØR K = 1 TØ N
180             PRINT S$(K); "," ;
185             NEXT K
190       PRINT
200       NEXT J
210 NEXT I
400 FØR I = 1 TØ N
410 PRINT S$(I)
420 NEXT I
900 DATA 6
910 DATA ZEB,JØHN,BILL,JACK,HAL,ABE
```

The RUN of B2Z shows just how the sorting program works. It starts with the list ZEB, JOHN, BILL, JACK, HAL, and ABE. It then goes through the whole list once (for I = 1) compares each word with the next word on the list and switches them if they are out of order. Thus for J = 1, the first word, ZEB, is switched with the second word, JØHN. This goes on until J = 5, at which the computer has compared every word on the list with its neighbor. The <u>last</u> word, ZEB, is now at the end of the list.

The second run (I = 2), only has to go up to J = 4, since we know the last word is at its correct place. (for every value of J, the words in the list have to be switched, since they are out of order. I = 3, J = 1 is an example where BILL and JACK are already in correct order, and so are not switched.) The third run only has to go up to J = 3, the fourth to J = 2, etc. The Ith run goes up to N - I.

Details of the Sorting Programs

(See Reference Manual pp. 10, 19, 21)

In NUM or in A2Z, line 5 is a dimension statement which reserves room in the computer's memory for the table of numbers (or strings). This is necessary for a list of more than 10 entries. Line 10 tells the computer exactly how many entries will be in the DATA statement in line 910.

Lines 20 - 40 are a loop which tells the computer to make up a table. (For example:

```
S$ (1) = "ZEB"
S$ (2) = "JØHN"
S$ (3) = "BILL"
S$ (4) = "JACK"
S$ (5) = "HAL"
S$ (6) = "ABE")
```

The FØR - TØ-NEXT are shorthand for a longer program that would do the same thing:

```
20 LET I = 1
30 READ S$(I)
40 IF I> = N THEN 100
50 LET I = I+1
60 GØTØ 20
```

Lines 40-210 are a pair of "nested loops" which tell the computer to go through the list N times (I = 1 TØ N). Each time it goes through the list (J = 1 TØ N - I), it compares each word with its neighbor, as in line 140. If the words need to be switched, lines 150-160 do the job. (The new variables X$ and Y$ are needed to act as a temporary storage for S$(J) and S$(J + 1) while they are being switched. If you didn't have this, you'd lose one of the words.)

Finally, lines 400-420 print out the results.

ØRD, A Program to Sort Two Lists at the Same Time

The following problem was handled with the computer. Arrange a list of foods both alphabetically and according to calorie values. Give both food and its calorie value.

The program ØRD does this, as shown below.

```
RUN


WLLØRD    12:17    M   THU 03/13/69

FØØD                CALØRIES

APPLE                72
BACØN                100
BANANA               100
BREAD, WHITE         60
BUTTER               100
CAKE, SLICE          100
CANDY BAR            297
CØØKIE               100
DØUGHNUT             213
EGG                  66
HAMBURGER            150
ICE CREAM            210
JUICE, ØRANGE        67
MILK, SKIM           84
MILK, WHØLE          165
ØKRA                 20
PEANUTS              84
PIE,APPLE            266
PØTATØ               100
SPINACH              20
SUGAR, TBSP.         52
TØMATØ               32
WATERMELØN           44

FØØD                CALØRIES

ØKRA                 20
SPINACH              20
TØMATØ               32
WATERMELØN           44
SUGAR, TBSP.         52
BREAD, WHITE         60
EGG                  66
JUICE, ØRANGE        67
APPLE                72
MILK, SKIM           84
PEANUTS              84
BACØN                100
BANANA               100
BUTTER               100
CAKE, SLICE          100
CØØKIE               100
PØTATØ               100
HAMBURGER            150
MILK, WHØLE          165
ICE CREAM            210
DØUGHNUT             213
PIE,APPLE            266
CANDY BAR            297

USED        6.17 UNITS.
```

```
LISTNH

5 DIM S(100), S$(100)
10 READ N
20 FØR I = 1 TØ N
30 READ S$(I), S(I)
40 NEXT I
100 FØR I = 1 TØ N
110      FØR J = 1 TØ N-I
120      LET X$ = S$(J)
130      LET Y$ = S$(J+1)
140      IF X$ <= Y$ THEN 200
150      LET S$(J) = Y$
160      LET S$(J+1) = X$
170      LET T = S(J)
180      LET S(J) = S(J+1)
190      LET S(J+1) = T
200      NEXT J
210 NEXT I
215 GØSUB 320
220 FØR I = 1 TØ N
225      FØR J = 1 TØ N-I
230      LET X = S(J)
235      LET Y = S(J+1)
240      IF X <= Y THEN 270
245      LET S(J) = Y
250      LET S(J+1) = X
255      LET T$ = S$(J)
260      LET S$(J) = S$(J+1)
265      LET S$(J+1) = T$
270      NEXT J
275 NEXT I
300 GØSUB 320
315 STØP
320 PRINT "FØØD", "CALØRIES"
330 PRINT
400 FØR I = 1 TØ N
410 PRINT S$(I), S(I)
420 NEXT I
425 PRINT
430 RETURN
900 DATA 23
910 DATA BUTTER,100,ICE CREAM,210,"MILK, SKIM",84
920 DATA "MILK, WHØLE",165,APPLE,72,BANANA,100
930 DATA "JUICE, ØRANGE",67,PEANUTS,84,"BREAD, WHITE",60
940 DATA "CAKE, SLICE",100,CØØKIE,100,DØUGHNUT,213
950 DATA "PIE,APPLE", 266,   "HAMBURGER", 150   ,"BACØN", 100
960 DATA EGG,66,CANDY BAR,297,"SUGAR, TBSP.",52
970 DATA "PØTATØ", 100,   "SPINACH", 20,   "ØKRA", 20,   "TØMATØ", 32
980 DATA "WATERMELØN", 44
999 END
```

ØRD is made up of A2Z and NUM. Lines 100-210 put the foods (S$(J)) in alphabetical order. Note that lines 170-190 are needed to rearrange the calories (S(J)) so they still go with the right food.

Lines 220-275 take the alphabetized lists and put the calories in correct order. Note that, within the set of foods with the same calorie value, the alphabetical order is kept.

Also note the use of the subroutine 320-430 in lines 215 and 300, where the same commands are needed twice in the same program. The subroutine saves the trouble of writing the same set of commands more than once.

The STØP command in line 315 is the same as GØTØ999.

PROBLEMS

1. Use A2Z to put a list of names in alphabetical order. Include two people with the same last name.
2. Use an $EDIT command to change A2Z into NUM.
3. Then use EDIT commands to extract lines 100-160 and make them into lines 220-270 of ØRD. Weave the two programs together.
4. Given a list of students with their room numbers. Make up a list of students by room, arranged alphabetically within the room. (Hint--only a few changes in ØRD are needed).
5. Given a list of English words with their equivalents in another language, write a program to make up an English-Foreign dictionary.

Note: In Mark I BASIC, strings can't be longer than 15 characters. Mark II BASIC will take longer strings. Strings can take up a whole line.

PRINT, A Program Which Illustrates the Print Command and Use of Subscripted Variables

Reference: BASIC LANGUAGE, Reference Manual (G.E. 202026A, 4-68), pages 23-24.

This is a simple program to illustrate the discussion of the PRINT command. Notice very carefully how both parts of the program work.

```
5 READ N
10 FOR M = 1 TO N
20       FOR J = 1 TO N
30       READ B (M,J)
35       PRINT B (M,J);
40       NEXT J
45       PRINT
50 NEXT M
60 FOR M = 1 TO N
70       FOR J = 1 TO M
80       PRINT B (M,J);
90       NEXT J
100      PRINT
110NEXT M
200 DATA 3
210 DATA 11,12,13,21,22,23,31,32,33
999 END

RUN


PRINT       10:40   M  TUE 10/08/68

 11     12     13
 21     22     23
 31     32     33
 11
 21     22
 31     32     33
```

```
QUADEQ     12:07     M     FRI     10/11/68

I'LL SØLVE THE QUADRATIC EQUATIØN AX^2 +BX+C = 0  IF YØU GIVE ME
THE NUMBERS A, B AND C.  WHEN YØU TYPE THEM, SEPARATE THEM BY
CØMMAS. THEN, REMEMBER TØ PRESS THE RETURN KEY. FØR EXAMPLE
WHAT ARE YØUR VALUES FØR A,B,C?  1,2,1 'RETURN'

WHAT ARE YØUR VALUES FØR A,B,C? 1,2,1
X1 = X2 = -1
DØ YØU WANT TØ DØ ANØTHER [TYPE YES ØR NØ]? YES

WHAT ARE YØUR VALUES FØR A,B,C? 1,2,2
X1 AND X2 ARE CØMPLEX; THE FIRST PART ØF THE PRINT
ØUT IS THE REAL PART, AND THE SECØND PART IS THE
IMAGINARY PART.

X1 = -1       1
X2 = -1      -1
DØ YØU WANT TØ DØ ANØTHER [TYPE YES ØR NØ]? YES

WHAT ARE YØUR VALUES FØR A,B,C? 1,5,4
X1 = -1
X2 = -4
DØ YØU WANT TØ DØ ANØTHER [TYPE YES ØR NØ]? NØ
IT'S BEEN MY PLEASURE TØ SERVE YØU. CØME AGAIN.


RAN     10.00 SEC
LISTNH

1   REM THIS PRØGRAM WAS WRITTEN BY DAVID HØWELL, CHM. MATH. DEPT.,"
2   REM RICHARD C. LEE HIGH SCHØØL, NEW HAVEN PUBLIC SCHØØLS."
3   PRINT
05  PRINT"
10  PRINT"I'LL SØLVE THE QUADRATIC EQUATIØN AX^2"+BX+C = 0 IF YØU GIVE ME"
11  PRINT"THE NUMBERS A, B AND C. WHEN YØU TYPE THEM, SEPARATE THEM BY"
12  PRINT"CØMMAS. THEN, REMEMBER TØ PRESS THE RETURN KEY. FØR EXAMPLE"
13  PRINT"WHAT ARE YØUR VALUES FØR A,B,C? 1,2,1 'RETURN'"
15  PRINT
16  PRINT"WHAT ARE YØUR VALUES FØR A,B,C";
20  INPUT A,B,C
50  IF A = 0 THEN 200
60  LET D = B↑2 - 4*A*C
70  IF D< 0 THEN 300
80  IF D = 0 THEN 400
100 LET X1 = (-B+SQR(D))/2/A
110 LET X2 = (-B-SQR(D))/2/A
120 PRINT "X1 = ";X1
121 PRINT "X2 = ";X2
123 GØ TØ 999
200 PRINT "NØT A QUADRATIC EQUATIØN."
201 GØ TØ 999
300 LET I = (SQR(-D))/2/A
310 LET R = -B/2/A
```

```
320 PRINT "X1 AND X2 ARE CØMPLEX;  THE FIRST PART ØF THE PRINT"
321 "ØUT IS THE REAL PART, AND THE SECØND PART IS THE"
322 PRINT "IMAGINARY PART."
323 PRINT
330 PRINT "X1 = ";R,I
340 PRINT "X2 = ";R,-I
350 GO TO 999
400 LET X1 = -B/2/A
410 PRINT "X1 = X2 = ";X1
999 PRINT "DØ YØU WANT TØ DØ ANØTHER (TYPE YES OR NØ)";
1000 INPUT A$
1001 IF A$ = "YES" THEN 15
1002 IF A$ = "NØ" THEN 1010
1003 GØTØ 999
1010 PRINT"IT'S BEEN MY PLEASURE TØ SERVE YØU.  CØME AGAIN."
1999 END
```

SHØES, A Program to Take Inventory for an Item that has Both Length and Width - Doubly Subscripted Variables, The TAB Command

In INV, we kept track of the number of pencils in a stockroom. Only <u>one</u> number needed to be given to give the hardness of pencils. Suppose you were a shoe store salesman, and needed to keep track of your stock. Since shoe sizes come in both length and width, you need <u>two</u> numbers. In computer language, you need to use doubly subscripted variables.

First, we have the computer type up a table of the number of shoes in stock, for each shoe size. -1 means that the size is not carried by the store. 0, 1, 2, ... give the number of pairs of shoes in stock. (Later on, we'll give the program that we used to print up this table.)

```
SHØES      13:09    M   WED 03/26/69

LENGTH 5.5    6  6.5    7   7.5    8   8.5    9   9.5    10  11    12   13
WIDTH
A       -1   -1   -1   -1    -1   -1    -1   -1     0     1   1    -1   -1
B       -1   -1   -1    0     0    1     1    0     2     2   2     1    0
C       -1   -1    1    2     2    1     0    2     3     2   1     0    1
D        0    1    1    2     3    3     2    1     2     1   1     1    0
E       -1   -1    1    1     0    1     0    1     0     1   1     1    0
EEE     -1   -1    0    1     0    1     0    1     0     1   1     1    0



USED        4.17 UNITS.
```

Next we, illustrate a RUN of SHØES. By checking the table, you'll see that the computer is a quick and accurate stock clerk. The LIST is given below.

```
SHOES      12:41   M  WED 03/26/69

TYPE LENGTH IN DECIMAL FORM, THEN A COMMA, THEN WIDTH. LENGTH
RUNS FROM 5.5 TO 10 IN HALF SIZES, 11,12, AND 13. WIDTH RUNS
FROMS A-E,EEE.
WHAT LENGTH, WIDTH? 5.5,A
WE DON'T CARRY SIZE 5.5  A

DO YOU WANT ANOTHER PAIR OF SHOES? 9.5,A

INPUT DATA NOT IN CORRECT FORMAT. RETYPE IT.
? YES
WHAT LENGTH, WIDTH? 9.5,A
SIZE 9.5  A IS OUT OF STOCK.

DO YOU WANT ANOTHER PAIR OF SHOES? YES
WHAT LENGTH, WIDTH? 10,A
 1 PAIRS, SIZE 10   A IN STOCK.

DO YOU WANT ANOTHER PAIR OF SHOES? YES
WHAT LENGTH, WIDTH? 5.5,EEE
WE DON'T CARRY SIZE 5.5  EEE

DO YOU WANT ANOTHER PAIR OF SHOES? YES
WHAT LENGTH, WIDTH? 13,EEE
SIZE 13   EEE IS OUT OF STOCK.

DO YOU WANT ANOTHER PAIR OF SHOES? YES
WHAT LENGTH, WIDTH? 12,EEE
 1 PAIRS, SIZE 12   EEE IN STOCK.

DO YOU WANT ANOTHER PAIR OF SHOES? YES
WHAT LENGTH, WIDTH? 9.5,D
 2 PAIRS, SIZE 9.5  D IN STOCK.

DO YOU WANT ANOTHER PAIR OF SHOES? NO



USED     27.17 UNITS.
```

The Program

In line.10, A DIM statement is given to make space for variables that run over 10 subscripts (See Learner's Guide, p. 21).

Lines 12-16 read in the list of widths from line 500 (A, B, C, D, E, EEE).

Lines 20-27 read in the list of lengths from line 510 (5, $5\frac{1}{2}$, 6,...10, 11, 12, 13). Lines 28-60 are a pair of "nested loops" which are needed to read in the above table of stock from lines 520-570.

Lines 65-80 are for putting in the shoe size.

Lines 100-140 tell the computer to search through its table to find the shoe size that was put in. If the computer finds it in the table, it goes to line 200. If it doesn't find the size, it prints as error message in line 150.

Line 200 decides whether the item is carried or not.

Line 230 decides whether the size is in stock or not. If it is, the computer goes to line 260 and gives the number of pairs of shoes in stock. If the stock is gone, the computer goes to line 240 and tells us the store is out of stock of our size. (Lines 230-250 could be omitted. Then, the computer merely would print "0 PAIRS, SIZE 10A IN STØCK", if the store were out of size 10A shoes.)

All branches eventually lead to line 300, where the salesman decides whether or not he wants to make another inquiry of the computer. If not, in line 340, the program ends.

```
LISTNH

10 DIM M(13,6), N(13)
12 FØR J = 1 TØ 6
14 READ MS(J)
16 NEXT J
20 FØR I = 1 TØ 13
25 READ N(I)
27 NEXT I
28 FØR J = 1 TØ 6
30      FØR I = 1 TØ 13
40      READ M(I,J)
50      NEXT I
60 NEXT J
65 PRINT "TYPE LENGTH IN DECIMAL FØRM, THEN A CØMMA, THEN WIDTH. LENGTH
70 PRINT "RUNS FRØM 5.5 TØ 10 IN HALF SIZES, 11,12, AND 13. WIDTH RUNS"
72 PRINT "FRØMS A-E,EEE."
75 PRINT "WHAT LENGTH, WIDTH";
80 INPUT N,MS
100 FØR I = 1 TØ 13
110 IF N = N(I) THEN 130
120 NEXT I
125 GØTØ 150
130 FØR J = 1 TØ 6
135 IF MS = MS(J) THEN 200
140 NEXT J
150 PRINT "YØUR SIZE IS NØT ØN THE STØCK LIST."
160 GØTØ 300
200 IF M(I,J)>= 0 THEN 230
210 PRINT "WE DØN'T CARRY SIZE"; N;MS
220 GØTØ 300
230 IF M(I,J)>0 THEN 260
240 PRINT"SIZE";N;MS;" IS ØUT ØF STØCK."
250 GØTØ 300
260 PRINT M(I,J); "PAIRS, SIZE";N;MS; " IN STØCK."
300 PRINT
310 PRINT "DØ YØU WANT ANØTHER PAIR ØF SHØES";
320 INPUT AS
330 IF AS = "YES" THEN 75
340 IF AS = "NØ" THEN 999
350 PRINT "PLEASE ANSWER YES ØR NØ. RETYPE YØUR ANSWER."
360 GØTØ 300
500 DATA A,B,C,D,E,EEE
510 DATA 5.5,6,6.5,7,7.5,8.,8.5,9,9.5,10,11,12,13
520 DATA -1,-1,-1,-1,-1,-1,-1,-1,0,1,1,-1,-1
530 DATA -1,-1,-1,0,0,1,1,0,2,2,2,1,0
540 DATA -1,-1,1,2,2,1,0,2,3,2,1,0,1
550 DATA 0,1,1,2,3,3,2,1,2,1,1,1,0
560 DATA-1,-1,1,1,0,1,0,1,0,1,1,1,0
570 DATA -1,-1,0,1,0,1,0,1,0,1,1,1,0
999 END
```

Extracting a Table - The TAB Command

To print up the table shown at the beginning of this paper, we could have written out the DATA in lines 510-570 and typed up the numbers by hand. To avoid the labor and possible human errors, a few lines were put in the program to print out the matrix of numbers automatically. This type of printing is very useful in "de-bugging" a program. Note how we used the EDIT LIST command (see <u>EDIT COMMANDS</u>, p. 6) to LIST only the lines of interest. Also note the use of the TAB command to print up a table with many columns in it. (See <u>Extensions</u>, p. 11) Of course, these lines could be added to print out an inventory of all sizes that the store carries.

```
EDIT LIST 62,400-490



62 GØTØ 400

400 PRINT "LENGTH";TAB(5);
405 FØR I = 1 TØ 13
410 PRINT N(I); TAB(5*I+5);
420 NEXT I
430 PRINT
435 PRINT "WIDTH"
440 FØR J = 1 TØ 6
445 PRINT MS(J);TAB(7);
450 FØR I = 1 TØ 13
460 PRINT M(I,J); TAB(5*I+7);
470 NEXT I
480 PRINT
490 NEXT J


USED        1.17 UNITS.
```

Problems: (Note, in doing these problems, you can save yourself labor by getting a tape of SHOES. If not, use a shorter table of shoe sizes.)

1 - Change a few lines in SHOES to write a data file to take the place of lines 520-570. Create a dummy file (see p. 16, Extensions), and write it during execution of your program.

2 - Change SHOES to keep track of the number of shoes bought or sold in each size, just as INV did. Check your program to make sure it works properly.

SPEEDY, A COMPUTER PROGRAM TO GIVE PERCENTAGES QUICKLY

Speedy is a very simple program which was designed for a basic algebra class which was strong on quick computations of simple arithmetic problems. To appeal to the class, and to illustrate the speed of the computer, the program was designed for fast input and quick data processing.

The data are fed in as a ratio of two numbers, the most common form of percentage problem. However, decimal input is allowed. The program does not handle mixed numbers (such as 1 + 2/3), which were excluded by the teacher on the grounds of his classroom experience.

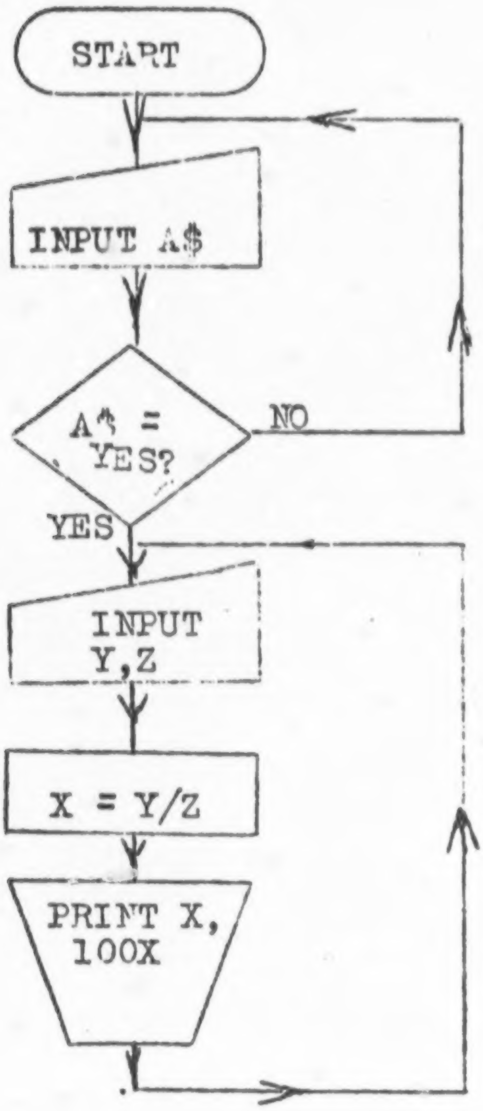
The results of the first classroom demonstration are shown in the print out. As always, all numbers and commands are typed by students. Their inexperience is shown in the entering of 4/3 for 3/4 and 5/4 for 4/5. However it is evident that by the end of the fifteen minute run, they had learned to enter reliable data.

The class was given a series of problems of gradually increasing difficulty. The students at the computer terminal pitted themselves against the rest of the class. As the problems became more difficult and the computer operators became more skillful, their inevitable triumph occurred. The tide turned at evaluating 185/200. After that, the opposition to the computer age collapsed. This makes an effective demonstration of the power of a computer.

The main part of the program (lines 200 - 220) finds the quotient of X/Y, rounds it to three decimal places and also expresses the result, rounded to the nearest tenth per cent. For speed and simplicity in running, the main part of the program is a simple, non-terminating loop. To break the loop and end the run, the operator types "STØP". The flow chart is shown with the demonstration run and list.

```
SPEEDY    10:14    M   THU 11/21/68

SPEEDY IS A CØMPUTER PRØGRAM TØ GIVE YØU PERCENTAGES QUICKLY.
AFTER THE '?', TYPE A NUMBER.  THEN PUSH THE 'RETURN' KEY.
I'LL GIVE YØUR NUMBER AS A DECIMAL FIRST, THEN AS A PERCENT.

    YØU CAN WRITE YØUR NUMBER IN ANY ØF THESE FØRMS:
FRACTIØN ØR RATIØ.  TØ WRITE 2/3, FIRST TYPE A 2, THEN A CØMMA,
THEN A THREE.  THEN PUSH THE 'RETURN' KEY, LIKE THIS:
? 2,3 'RETURN'

TØ WRITE A DECIMAL ØR WHØLE NUMBER, TYPE THE NUMBER, A CØMMA,
AND THEN A ØNE.  TØ WRITE 1.75, TYPE 1.75,1 'RETURN'
DØ NØT TYPE MIXED NUMBERS LIKE 1 2/3.  I WØN'T ACCEPT THEM.

REMEMBER, WHEN YØU WANT TØ STØP, JUST TYPE 'STOP'.

ARE YØU READY [PRINT YES WHEN YØU ARE]? YES
? 1,2
 .5     =  50      PER CENT
? 1,3
 .333   =  33.3    PER CENT
? 4,3
 1.333     =  133.3     PER CENT
? 3,4
 .75    =  75      PER CENT
? 5,4
 1.25   =  125     PER CENT
? 4,5
 .8     =  80      PER CENT
? 19,20
 .95    =  95      PER CENT
? 185,200
 .925   =  92.5    PER CENT
? 2336,6183
 .378   =  37.8    PER CENT
? 27232.  ,65876
 .413   =  41.3    PER CENT
```

```
? 127232,265876
 .479  =  47.9  PER CENT
? STØP

RAN      22.00 SEC
BYE

*** ØFF AT 10:28  ELAPSED TERMINAL TIME =     15 MIN.

LISTNH

10 PRINT "SPEEDY IS A CØMPUTER PROGRAM TØ GIVE YØU PERCENTAGES QUICKLY."
20 PRINT "AFTER THE '?', TYPE A NUMBER.  THEN PUSH THE 'RETURN' KEY."
30 PRINT " I'LL GIVE YØUR NUMBER AS A DECIMAL FIRST, THEN AS A PERCENT."
40 PRINT
50 PRINT "     YØU CAN WRITE YØUR NUMBER IN ANY ØF THESE FØRMS:"
60 PRINT "FRACTIØN ØR RATIØ.  TØ WRITE 2/3, FIRST TYPE A 2, THEN A CØMMA,"
70 PRINT "THEN A THREE. THEN PUSH THE 'RETURN' KEY, LIKE THIS:"
80 PRINT "? 2,3 'RETURN' "
90 PRINT
100 PRINT "TØ WRITE A DECIMAL ØR WHØLE NUMBER, TYPE THE NUMBER,A COMMA,"
110 PRINT "AND THEN A ØNE. TØ WRITE 1.75, TYPE 1.75,1 'RETURN'"
115 PRINT "DØ NØT TYPE MIXED NUMBERS LIKE 1 2/3. I WØN'T ACCEPT THEM."
116 PRINT
120 PRINT "REMEMBER, WHEN YØU WANT TØ STØP, JUST TYPE 'STØP'."
130 PRINT
140 PRINT "ARE YØU READY (PRINT YES WHEN YØU ARE)";
150 INPUT A$
160 IF A$ = "YES" THEN 200
170 PRINT "Ø.K., WHEN YØU'RE READY YØU JUST LET ME KNØW."
180 GØTØ 140
200 INPUT Y,Z
205 LET X = Y/Z
210 PRINT INT(1000*X+.5)/1000; " = "; INT(1000*X+.5)/10; " PER CENT"
220 GØTØ 200
999 END
```


START
INPUT A$
A$ = YES?
NO
YES
INPUT Y,Z
X = Y/Z
PRINT X, 100X

FLOW CHART for SPEEDY

TAKHØM, a Basic Program to Make up a Paycheck. Data Banks

TAKHØM is a program which was used to demonstrate "consumer math" in a second year, terminal mathematics course for tenth grade pupils. TAKHØM is an example of a "data bank" which stores a large amount of information in the computer's memory. The computer looks up the information whenever it is needed.

To make up a paycheck, one has to know the deductions for income tax and social security (F.I.C.A.). The textbook for the course gave separate tables for these taxes.

INCOME TAX WITHOLDING TABLE

If the payroll period with respect to an employee is Weekly:

And the wages are--		And the number of witholding exemptions claimed is--										
At least	But less than	0	1	2	3	4	5	6	7	8	9	10 or more
		The amount of income tax to be witheld shall be --										
80	82	11.30	9.50	7.80	6.00	4.20	2.40	.60	0	0	0	0
82	84	11.60	9.80	8.00	6.20	4.40	2.60	.90	0	0	0	0
84	86	11.90	10.10	8.30	6.50	4.70	2.90	1.10	0	0	0	0
86	88	12.20	10.40	8.60	6.80	5.00	3.20	1.40	0	0	0	0
88	90	12.50	10.70	8.90	7.10	5.30	3.50	1.70	0	0	0	0
90	92	12.70	10.90	9.20	7.40	5.60	3.80	2.00	.20	0	0	0
92	94	13.00	11.20	9.40	7.60	5.80	4.00	2.30	.50	0	0	0
94	96	13.30	11.50	9.70	7.90	6.10	4.30	2.50	.70	0	0	0
96	98	13.60	11.80	10.00	8.20	6.40	4.60	2.80	1.00	0	0	0
98	100	13.90	12.10	10.30	8.50	6.70	4.90	3.10	1.30	0	0	0

SOCIAL SECURITY EMPLOYEE TAX TABLE - $3\frac{5}{8}$ PERCENT

Wages		Social Security tax to be withheld	Wages		Social Security tax to be withheld	Wages		Social Security tax to be withheld
At least	But less than		At least	But less than		At least	But less than	
$84.00	$84.28	$3.05	$89.52	$89.80	$3.25	$97.80	$98.07	$3.55
84.28	84.56	3.06	89.80	90.07	3.26	98.07	98.35	3.56
84.56	84.83	3.07	90.07	90.35	3.27	98.35	98.63	3.57
84.83	85.11	3.08	90.35	90.63	3.28	98.63	98.90	3.58
85.11	85.38	3.09	90.63	90.90	3.29	98.90	99.18	3.59
85.38	85.66	3.10	90.90	91.18	3.30	99.18	99.45	3.60
85.66	85.94	3.11	91.18	91.45	3.31	99.45	99.73	3.61
85.94	86.21	3.12	91.45	91.73	3.32	99.73	100.00	3.62
86.21	86.49	3.13	91.73	92.00	3.33	100.00	100.28	3.63
86.49	86.76	3.14	92.00	92.28	3.34	100.28	100.56	3.64

The entire tax table is stored in the computer as DATA. When the number of exemptions and the gross wages are given, the computer computes the correct location in the table and takes out the correct deduction. The F.I.C.A. tax is computed directly without the use of a table.

A sample problem is shown, along with the list of the program. Note that the computer prints the wages, each deduction, total deductions and take home pay.

```
TAKHØM    13:59  .M  MØN 11/18/68

THIS IS A COMPUTER PRØGRAM WHICH USES THE WITHHØLDING AND
SØCIAL SECURITY TABLES TØ FIGURE ØUT THE TAKE-HØME PAY.
THE TABLES ØNLY ARE GØØD FØR WAGES BETWEEN $80-100 PER WEEK.

WHEN I ASK YOU , TYPE THE TØTAL WAGES IN DØLLARS [LEAVE ØUT
THE $] AND PUSH THE RETURN KEY. FØR EXAMPLE,
WHAT ARE THE TØTAL WAGES? 84.20 'RETURN'

WHAT ARE THE TØTAL WAGES? 85.90
HØW MANY WITHHØLDING EXEMPTIØNS ARE CLAIMED? 2
WITH  2  EXEMPTIØNS AND TØTAL WAGES ØF $ 85.9

THE DEDUCTIØNS ARE:
INCØME TAX          $ 8.3
F.I.C.A. TAX        $ 3.11
TØTAL DEDUCTIØNS    $ 11.41

NET WAGES $ 74.49    THIS IS YOUR TAKE-HØME PAY.

DØ YØU WANT TØ RUN AGAIN [PLEASE ANSWER YES ØR NØ]? NØ

I'VE ENJØYED SERVING YOU. CØME AGAIN. NØW SAY 'GØØDBYE' TØ ME.




RAN      7.83 SEC
```

```
5 REM READ TAX TABLE FRØM PIPER + GRUBER,'APPLIED BUSINESS MATHEMATICS'
6 REM (SØUTHWESTERN PUBLISHING CØ, 1965), P. 101
9 DIM A(10,11)
10 MAT READ A
20 PRINT "THIS IS A CØMPUTER PRØGRAM WHICH USES THE WITHHØLDING AND"
30 PRINT "SØCIAL SECURITY TABLES TØ FIGURE ØUT THE TAKE-HØME PAY."
40 PRINT "THE TABLES ØNLY ARE GØØD FØR WAGES BETWEEN $80-100 PER WEEK."
50 PRINT
55 PRINT "WHEN I ASK YØU , TYPE THE TØTAL WAGES IN DØLLARS (LEAVE ØUT"
56 PRINT "THE $) AND PUSH THE RETURN KEY. FØR EXAMPLE,"
57 PRINT "WHAT ARE THE TØTAL WAGES? 84.20 'RETURN'"
60 PRINT
70 PRINT "WHAT ARE THE TØTAL WAGES";
80 INPUT W
90 PRINT "HØW MANY WITHHØLDING EXEMPTIØNS ARE CLAIMED";
100 INPUT J
102 IF J < 10 THEN 106
103 LET J = 10
105 REM FIND ØUT CØRRECT LINE ØF TABLE
106 LET J = J+1
110 IF W < 80 THEN 140
120 IF W >= 100 THEN 140
130 GØTØ 150
140 PRINT "SØRRY, YØUR WAGES ARE NØT IN MY TABLES. SEE YØUR FRIENDLY"
145 PRINT "I.R.S. AGENT, PLEASE."
147 GØTØ 250
150 LET I = INT(W/2) - 39
151 REM FIND F, THE FICA TAX; T, THE TØTAL DEDUCTIØN, + N, NET WAGES.
155 LET F = INT(3.625*W + .5)/100
160 LET T =F + A(I,J)
165 LET N = W-T
170 PRINT "WITH "; J-1 ;" EXEMPTIØNS AND TØTAL WAGES ØF $";W
180 PRINT
190 PRINT "THE DEDUCTIØNS ARE:"
200 PRINT "INCØME TAX","   $";A(I,J)
250 PRINT "F.I.C.A. TAX","   $";F
220 PRINT "TØTAL DEDUCTIØNS  $";T
230 PRINT
240 PRINT "NET WAGES $";N;" THIS IS YØUR TAKE-HØME PAY."
250 PRINT
260 PRINT "DØ YØU WANT TØ RUN AGAIN (PLEASE ANSWER YES ØR NØ)";
270 INPUT A$
275 PRINT
280 IF A$ = "YES" THEN 70
290 IF A$ = "NØ" THEN 320
300 PRINT "SØRRY, I DIDN'T UNDERSTAND YØU."
310 GØTØ 260
320 PRINT"I'VE ENJØYED SERVING YØU. CØME AGAIN. NØW SAY 'GØØDBYE' TØ ME."
399 REM EACH LINE ØF DATA IS ØNE LINE ØF THE WITHHØLDING TABLE
400 DATA 11.30,9.5,7.8,6.0,4.2,2.4,.6,0,0,0,0
401 DATA 11.60,9.8,8.0,6.2,4.4,2.6,.9,0,0,0,0
402 DATA 11.9,10.1,8.3,6.5,4.7,2.9,1.1,0,0,0,0
403 DATA 12.2,10.4,8.6,6.3,5,3.2,1.4,0,0,0,0
404 DATA 12.5,10.7,8.9,7.1,5.3,3.5,1.7,0.0,0,0,0
405 DATA 12.7,10.9,9.2,7.4,5.6,3.8,2.0,.2,0,0,0
406 DATA 13,11.2,9.4,7.6,5.8,4,2.3,.5,0,0,0
407 DATA 13.3,11.5,9.7,7.9,6.1,4.3,2.5,.7,0,0,0
408 DATA 13.6,11.8,10.0,8.2,6.4,4.6,2.8,1.0,0,0,0
409 DATA 13.9,12.1,10.3,8.5,6.7,4.9,3.1,1.3,0,0,0
```

The Program - DATA in Matrix Form

The structure of the program can be seen from the list. Since the flow is essentially a simple, linear one, with no branches, the flowchart is omitted.

The unique feature of the program is the storage of data in the form of a matrix, which has 10 rows and 11 columns in the present case. The dimension statement in line 9 and the MAT READ statement in line 10 are all that is needed to read in the DATA.

During the run, the gross wages, W, are typed in (line 80). The number of exemptions, J, is used to find the correct column of the table (lines 100-106). Lines 110 and 120 screen out wages not included in the table. Line 150 computes the correct line, I, in the tax table. Line 155 computes the F.I.C.A. tax, F, by taking the correct percentage of the pay and rounding off to the nearest penny. Line 160 computes the total deductions, T, by adding F to the tax, A(I,J), obtained from the tax table. Lines 190-240 print out all this information, plus the take home pay.

Up to date rates can be obtained by changing lines 155, 400-409 of the program.

Problems:

1. Get a copy of the new tax rates and bring TAKHØM up to date.
2. Rewrite the program to keep a running total of F.I.C.A. and withholding taxes. (Enter last month's total and add it to the deductions for this month). Write the program to insure that the F.I.C.A. tax does not exceed the maximum amount. (At present the rate is 4.8% of the first $7,800.)

TEMP, A Program To Give Temperature In All Three Scales.

The conversion among Celsius (centigrade), Fahrenheit and Kelvin (absolute) scales involves the simple linear formulae:

$$^{o}K = {^{o}C} + 273.16$$

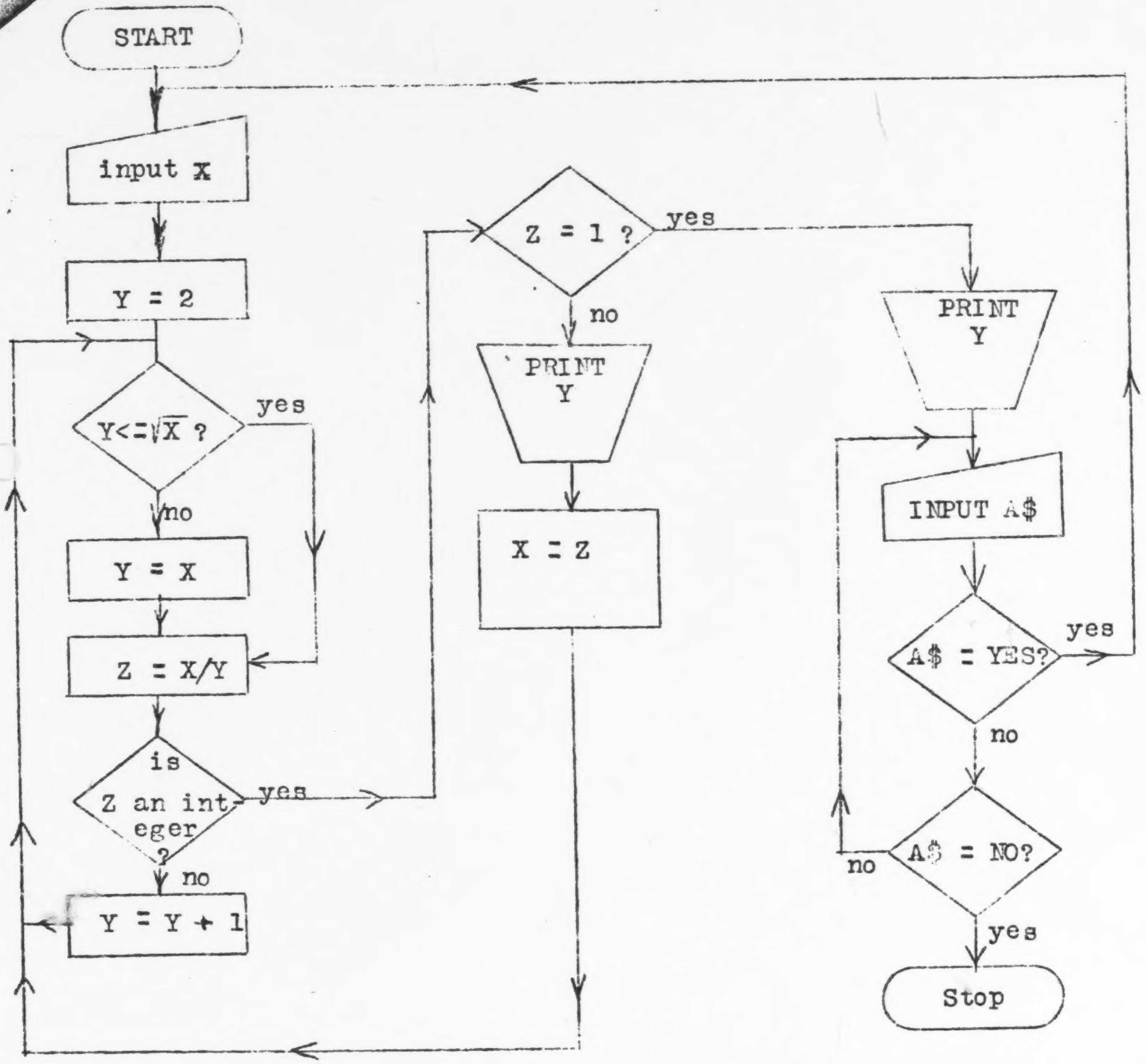
$$^{o}F = (9/5)(^{o}C) + 32\ .$$

This program permits input of any one of the three temperatures and solves for the other two. Since computers cannot solve even the simplest linear equations, it is necessary to write out all three sets of solutions, which correspond to each scale of temperature as input. These are given in lines 200-220, 250-270 and 300-310. The results are rounded to one decimal place in lines 350-370. Temperatures below absolute zero are rejected by the computer with an explanation in lines 380 and 600. As is shown in the classroom run, this gave the teacher a chance to go into the absolute temperature scale. This program was used in ninth grade physical science class. It could be used wherever the relationship between the temperature scales is being studied. It appeared to be most effective in the brighter classes. Scott Cutler, a Lee student, wrote this program.

```
TEMP      9:40     M   TUE 11/26/68

THIS CØMPUTER PRØGRAM WILL GIVE A TEMPERATURE IN ALL THREE
TEMPERATURE SCALES: FAHRENHEIT, CELSIUS (CENTIGRADE), AND
  LVIN (ABSØLUTE).

     TYPE THE TEMPERATURE, THEN A CØMMA, AND THEN YØUR UNITS:
TYPE 'F' FØR FAHRENHEIT, TYPE 'C' FØR CELSIUS (CENTIGRADE),
ØR 'K' FØR KELVIN (ABSØLUTE).  THEN PRESS THE 'RETURN' KEY.
FØR EXAMPLE, AFTER I ASK YØU:
WHAT ARE THE TEMPERATURE, UNITS? (YØU TYPE) 98.6,F 'RETURN'
GØT IT? NØW I'LL ASK YØU FØR YØUR TEMPERATURE.

WHAT ARE THE TEMPERATURE, UNITS? 78,C

TEMP. KELVIN= 351.2  TEMP. FAHRENHEIT= 172.4  TEMP. CENTIGRADE= 78
CAN I HELP YØU AGAIN (PLEASE ANSWER YES ØR NØ)? YES

WHAT ARE THE TEMPERATURE, UNITS? -300,C
```

```
IMPØSSIBLE. THE TEMPERATURE CAN'T BE LESS THAN ABSØLUTE ZERØ.
CAN I HELP YØU AGAIN (PLEASE ANSWER YES ØR NØ)? YES

WHAT ARE THE TEMPERATURE, UNITS? 0,K

TEMP. KELVIN= 0 TEMP. FAHRENHEIT=-459.7  TEMP. CENTIGRADE=-273.2
CAN I HELP YØU AGAIN (PLEASE ANSWER YES ØR NØ)? ND
WHAT? PLEASE READ THE INSTRUCTIØNS THIS TIME.
CAN I HELP YØU AGAIN (PLEASE ANSWER YES ØR NO)? NØ

   'S BEEN FUN. LET'S HAVE ANØTHER CØMPUTER SESSION AGAIN.
NØW SAY GØØDBYE TO ME. GØØDBYE.


RAN     12.83 SEC
GØØDBYE

*** ØFF AT  9:52  ELAPSED TERMINAL TIME =     15 MIN.

LISTNH


10 PRINT "THIS CØMPUTER PRØGRAM WILL GIVE A TEMPERATURE IN ALL THREE"
11 PRINT "TEMPERATURE SCALES: FAHRENHEIT, CELSIUS (CENTIGRADE),AND"
12 PRINT "KELVIN (ABSØLUTE)."
15 PRINT
20 PRINT "     TYPE THE TEMPERATURE, THEN A CØMMA, AND THEN YØUR UNITS:"
25 PRINT "TYPE 'F' FØR FAHRENHEIT, TYPE 'C' FØR CELSIUS (CENTIGRADE),"
26 PRINT "OR 'K' FØR KELVIN (ABSØLUTE). THEN PRESS THE 'RETURN' KEY."
27 PRINT "FØR EXAMPLE, AFTER I ASK YØU:"
28 PRINT "WHAT ARE THE TEMPERATURE, UNITS? (YØU TYPE) 98.6,F 'RETURN' "
29 PRINT "GØT IT? NØW I'LL ASK YØU FØR YØUR TEMPERATURE. "
   PRINT
31 PRINT "WHAT ARE THE TEMPERATURE, UNITS";
32 INPUT A, B$
35 PRINT
40 IF B$= "F" THEN 200
50 IF B$= "K" THEN 250
60 IF B$= "C" THEN 300
70 PRINT "SØRRY, I DØN'T UNDERSTAND YØU. PLEASE READ INSTRUCTIØNS."
80 GØTØ 20
200 LET C=5*(A-32)/9
210 LET K=C+273.16
220 LET F=A
225 GØTØ 350
250 LET C=A-273.16
260 LET = 32 + C*9/5
```

```
270 LET K=A
275 GØTØ 350
300 LET F = 32 + A*9/5
302 LET C = A
310 LET K=C+273.16
325 GØTØ 350
350 LET X=INT(10*F+.5)/10
360 LET Y = INT(10*K+.5)/10
370 LET Z=INT(10*C+.5)/10
380 IF K<0 THEN 600
400 PRINT "TEMP. KELVIN=";"TEMP. FAHRENHEIT=";X;"TEMP. CENTIGRADE=";Z
410 GØTØ 610
600 PRINT "IMPØSSIBLE. THE TEMPERATURE CAN'T BE LESS THAN ABSØLUTE ZERØ."
610 PRINT "CAN I HELP YØU AGAIN (PLEASE ANSWER YES OR NØ)";
615 INPUT C$
620 IF C$="YES" THEN 650
630 IF C$ = "NØ" THEN 655
635 PRINT "WHAT? PLEASE READ THE INSTRUCTIØNS THIS TIME."
636 GØTØ 610
650 GØTØ 30
655 PRINT
660 PRINT "IT'S BEEN FUN. LET'S HAVE ANØTHER CØMPUTER SESSIØN AGAIN."
670 PRINT "NØW SAY GØØDBYE TØ ME. GØØDBYE."
999 END
```


START
input X
Y = 2
Y<=√X ?
yes
no
Y = X
Z = X/Y
is Z an integer ?
yes
no
Y = Y + 1
Z = 1 ?
yes
no
PRINT Y
X = Z
PRINT Y
INPUT A$
A$ = YES?
yes
no
A$ = NO?
no
yes
Stop

FLOW CHART for FACTØR

Trigonometric Programs - RTTRI and TRIANG***-

Defined Functions

These two programs are useful demonstrations of trigonometric solutions of a triangle. RTTRI is specifically written for use in elementary trigonometry classes, in which two parts of a right triangle are given. TRIANG*** is a remarkably general program which will handle any case that arises. TRIANG*** is stored in the Mark I computer. It is a library program of G.E.

The list and sample runs of both programs are given. In RTTRI, a 30° - 60° - 90° triangle is given. In TRIANG***, the triangle can be either 30° - 105° - 45° or 30° - 15° - 135°, depending on which of two possible solutions is chosen.

Note that both programs make liberal use of subroutines to keep the length of the list under control. Also worth noting is the use of the DEF FN in line 105 of TRIANG***. This allows defining a function before its arguments are specified. FNS (I) is used in line 720, for giving angles rounded to the nearest hundredth of a second of arc.

```
RUN



RTTRI      13:40    M   WED 02/26/69

I'LL GIVE YOU ALL THE PARTS OF A RIGHT TRIANGLE, IF YOU GIVE
ME ANY TWO PARTS: HYPOTENUSE AND A SIDE (HS), HYPOTENUSE AND
AN ANGLE (HA), ANGLE AND OPPOSITE SIDE (AS1), ANGLE AND
ADJACENT SIDE (AS2), OR TWO SIDES (SS). DON'T FORGET TO PRESS
THE 'RETURN' KEY, AFTER YOU HAVE TYPED YOUR MESSAGE.

WHAT IS GIVEN (HS,HA,AS1,AS2, OR SS)? HS

WHAT ARE YOUR VALUES FOR THE HYPOTENUSE, SIDE
(TYPE THEM WITH A COMMA BETWEEN THEM)? 1,.5

OPPOSITE SIDE = .5              ADJACENT ANGLE:  30   DEG      0 MIN.
ADJACENT SIDE = .866025         OPPOSITE ANGLE:  60   DEG      0 MIN.
   HYPOTENUSE = 1.

DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)? YES

WHAT IS GIVEN (HS,HA,AS1,AS2, OR SS)? HA

WHAT ARE YOUR VALUES FOR THE HYPOTENUSE, ANGLE (DEGREES)
(TYPE THEM WITH A COMMA BETWEEN THEM)? 1,30

OPPOSITE SIDE = .5              ADJACENT ANGLE:  30   DEG      0 MIN.
ADJACENT SIDE = .866025         OPPOSITE ANGLE:  60   DEG      0 MIN.
   HYPOTENUSE = 1.

DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)? AS1
SORRY, I DIDN'T UNDERSTAND YOU. PLEASE FOLLOW INSTRUCTIONS.

DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)? YES

WHAT IS GIVEN (HS,HA,AS1,AS2, OR SS)? AS1

WHAT ARE YOUR VALUES FOR THE ANGLE (DEG), OPPOSITE SIDE
(TYPE THEM WITH A COMMA BETWEEN THEM)? 30,.5

OPPOSITE SIDE = .5              ADJACENT ANGLE:  30   DEG      0 MIN.
ADJACENT SIDE = .866025         OPPOSITE ANGLE:  60   DEG      0 MIN.
   HYPOTENUSE = 1.

DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)? YES

WHAT IS GIVEN (HS,HA,AS1,AS2, OR SS)? AS2

WHAT ARE YOUR VALUES FOR THE ANGLE (DEG), ADJACENT SIDE
(TYPE THEM WITH A COMMA BETWEEN THEM)? 30,.866025

OPPOSITE SIDE = .5              ADJACENT ANGLE:  30   DEG      0 MIN.
ADJACENT SIDE = .866025         OPPOSITE ANGLE:  60   DEG      0 MIN.
   HYPOTENUSE = 1.
```

```
DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)? YES

WHAT IS GIVEN (HS,HA,AS1,AS2, OR SS)? SS

WHAT ARE YOUR VALUES FOR THE TWO SIDES
(TYPE THEM WITH A COMMA BETWEEN THEM)? .5,.866025

OPPOSITE SIDE = .5               ADJACENT ANGLE:  30   DEG       0 MIN.
ADJACENT SIDE = .866025          OPPOSITE ANGLE:  60   DEG       0 MIN.
   HYPOTENUSE = 1.

DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)? NO
IT'S BEEN FUN. COME AGAIN. GOODBYE. NOW SAY GOODBYE TO ME.



USED      29.83 UNITS.
```

```
LISTNH

20 PRINT "I'LL GIVE YOU ALL THE PARTS OF A RIGHT TRIANGLE, IF YOU GIVE"
30 PRINT "ME ANY TWO PARTS: HYPOTENUSE AND A SIDE (HS), HYPOTENUSE AND"
40 PRINT "AN ANGLE (HA), ANGLE AND OPPOSITE SIDE (AS1), ANGLE AND"
50 PRINT "ADJACENT SIDE (AS2), OR TWO SIDES (SS). DON'T FORGET TO PRESS"
60 PRINT "THE 'RETURN' KEY, AFTER YOU HAVE TYPED YOUR MESSAGE."
70 PRINT
80 PRINT "WHAT IS GIVEN (HS,HA,AS1,AS2, OR SS)";
90 INPUT A$
100 IF A$ = "SS" THEN 200
110 IF A$ = "HS" THEN 400
120 IF A$ = "HA" THEN 500
130 IF A$ = "AS1" THEN 600
140 IF A$ = "AS2" THEN 700
150 GOSUB 990
160 GOTO 70
200 LET X$ = "TWO "
210 LET Y$ = "SIDES"
220 GOSUB 900
230 INPUT S(1), S(2)
240 LET C$(2) = B$(1) = "OPPOSITE"
250 LET C$(1) = B$(2) = "ADJACENT"
260 LET A(1) = ATN(S(1)/S(2))
270 LET A(2) = 1.57079632 - A(1)
280 LET H = SQR(S(1)↑2+S(2)↑2)
285 PRINT
290 FOR I = 1 TO 2
300 LET B(I) = 57.2957795*A(I)
310 LET D(I) = INT(B(I))
320 LET M(I) = INT(60*(B(I)-D(I))+.5)
325 IF M(I)<60 THEN 330
326 LET M(I) =0
327 LET D(I) = D(I)+1
330 PRINT B$(I);" SIDE =";S(I),C$(I);" ANGLE: ";D(I);"DEG ",M(I);"MIN."
340 NEXT I
350 PRINT "   HYPOTENUSE =";H
360 PRINT
370 PRINT "DO YOU WANT TO DO ANOTHER PROBLEM (ANSWER YES OR NO)";
380 INPUT E$
390 IF E$ = "YES" THEN 70
392 IF E$ = "NO" THEN 998
394 GOSUB 990
395 GOTO 360
400 LET X$ = "HYPOTENUSE, "
410 LET Y$ = "SIDE"
420 GOSUB 900
430 INPUT H, S(1)
440 IF H>= S(1) THEN 470
450 PRINT "IMPOSSIBLE. THE SIDE CAN'T BE LONGER THAN THE HYPOTENUSE."
460 GOTO 360
470 LET S(2) = SQR(H↑2-S(1)↑2)
480 GOTO 240
500 LET X$ = "HYPOTENUSE, "
510 LET Y$ = "ANGLE (DEGREES)"
520 GOSUB 900
530 INPUT H,A
535 GOSUB 950
540 LET S(1) = H*SIN(A)
550 LET S(2) = H*COS(A)
560 GOTO 240
```

```
600 LET X$ = "ANGLE (DEG), "
610 LET Y$ = "OPPOSITE SIDE"
620 GOSUB 900
630 INPUT A, S(1)
640 GOSUB 950
650 LET S(2) = S(1)/TAN(A)
660 GOTO 240
700 LET X$ = "ANGLE (DEG), "
710 LET Y$ = "ADJACENT SIDE"
720 GOSUB 900
730 INPUT A, S(2)
740 GOSUB 950
750 LET S(1) = S(2)*TAN(A)
760 GOTO 240
810 RETURN
900 PRINT
910 PRINT "WHAT ARE YOUR VALUES FOR THE ";X$;Y$
920 PRINT "(TYPE THEM WITH A COMMA BETWEEN THEM)";
930 RETURN
950 IF ABS(A-45)<45 THEN 975
960 PRINT "THE ANGLE MUST BE BETWEEN 0 AND 90 DEGREES."
970 GOTO 360
975 LET A = A/57.2957795
980 RETURN
990 PRINT "SORRY, I DIDN'T UNDERSTAND YOU. PLEASE FOLLOW INSTRUCTIONS."
995 RETURN
998 PRINT "IT'S BEEN FUN. COME AGAIN. GOODBYE. NOW SAY GOODBYE TO ME."
999 END
```

```
TRIANG     10:13    M  WED 02/26/69


THIS PROGRAM WILL FIND THE UNKNOWN FEATURES OF ANY
TRIANGLE, GIVEN ONE SIDE AND ANY TWO OTHER PARTS.


WHAT WILL BE GIVEN (1=SSS, 2=SAS, 3=SSA, 4=ASA, 5=AAS)? 3

NOTE: SPECIFY ANGLES AS 'DEGREES, MINUTES, SECONDS'  OR
      'DEGREES, MINUTES.DECIMAL, 0' (I.E., SECONDS=0)

WHAT ARE SIDE1, SIDE3, ANGLE1? .707,1.0,30,0,0

ONE SOLUTION IS:


                    1                  2                  3
                 -------            -------            -------

SIDE             .707               1.36587            1

ANGLE (RAD)      .523599            1.83244            .785549

ANGLE (DEG)      29                 104                45
      (MIN)      59                 59                 0
      (SEC)      59.98              28.84              31.17

ALT TO SIDE      .965965            .5                 .682937


RADIUS OF CIRCUMSCR CIRCLE =  .707
RADIUS OF INSCRIBED CIRCLE =  .222247
          AREA OF TRIANGLE =  .341469


A SECOND SOLUTION IS:


SIDE             .707               .366177            1

ANGLE (RAD)      .523599            .261951            2.35604

ANGLE (DEG)      30                 15                 134
      (MIN)      0                  0                  59
      (SEC)      .08                31.23              28.67

ALT TO SIDE      .258965            .5                 .183088


RADIUS OF CIRCUMSCR CIRCLE =  .707
RADIUS OF INSCRIBED CIRCLE =  .088313
          AREA OF TRIANGLE =  9.1[illegible]442[illegible]-02



ANOTHER CASE (1=YES)? 2
```

```
USED      11.50 UNITS.
```

```
100 PRINT
105 DEF FNS(I)= .01*INT(6000*(B(I)-INT(B(I)))+.5)
110 PRINT "THIS PROGRAM WILL FIND THE UNKNOWN FEATURES OF ANY"
115 PRINT "TRIANGLE, GIVEN ONE SIDE AND ANY TWO OTHER PARTS."
120 PRINT
125 PRINT
130 PRINT "WHAT WILL BE GIVEN (1=SSS, 2=SAS, 3=SSA, 4=ASA, 5=AAS)";
135 INPUT N
140 PRINT
145 LET R8=0
150 LET R9=0
155 IF (N-1)*(N-2)*(N-3)*(N-4)*(N-5)=0 THEN 170
160 PRINT "ANSWER 1, 2, 3, 4, OR 5 PLEASE..."
165 GOTO 125
170 IF N=1 THEN 210
175 PRINT "NOTE: SPECIFY ANGLES AS 'DEGREES, MINUTES, SECONDS' OR"
180 PRINT "      'DEGREES, MINUTES.DECIMAL, 0' (I.E., SECONDS=0)"
185 PRINT
190 IF N=2 THEN 230
195 IF N=4 THEN 270
200 IF N=5 THEN 335
205 IF N=3 THEN 385
210 PRINT "WHAT ARE SIDE1, SIDE2, SIDE3";
215 INPUT S(1),S(2),S(3)
220 GOSUB 540
225 GOTO 630
230 PRINT "WHAT ARE SIDE1, ANGLE3, SIDE2";
235 INPUT S(1),A(3),B(3),C(3),S(2)
240 GOSUB 830
245 LET I=3
250 GOSUB 620
255 LET S(3)=SQR(S(1)↑2+S(2)↑2-2*S(1)*S(2)*COS(R(3)))
260 GOSUB 540
265 GOTO 630
270 PRINT "WHAT ARE ANGLE1, SIDE3, ANGLE2";
275 INPUT A(1),B(1),C(1),S(3),A(2),B(2),C(2)
280 GOSUB 830
285 FOR I=1 TO 2
290 GOSUB 620
295 NEXT I
300 LET R(3)=3.14159265-R(1)-R(2)
305 LET I=3
310 GOSUB 575
315 FOR I=1 TO 2
320 GOSUB 530
325 NEXT I
330 GOTO 630
335 PRINT "WHAT ARE ANGLE3, ANGLE1, SIDE3";
340 INPUT A(3),B(3),C(3),A(1),B(1),C(1),S(3)
345 GOSUB 830
350 FOR I=1 TO 3 STEP 2
355 GOSUB 620
360 NEXT I
365 LET R(2)=3.14159265-R(1)-R(3)
370 LET I=2
375 GOSUB 575
380 GOTO 315
385 PRINT "WHAT ARE SIDE1, SIDE3, ANGLE1";
390 INPUT S(1),S(3),A(1),B(1),C(1)
395 GOSUB 830
```

```
670 PRINT
675 PRINT " ";"   1","   2","   3"
680 PRINT " ","--------","--------","--------"
685 PRINT
690 PRINT "SIDE",S(1),S(2),S(3)
695 PRINT
700 PRINT "ANGLE (RAD)",R(1),R(2),R(3)
705 PRINT
710 PRINT "ANGLE (DEG)",A(1),A(2),A(3)
715 PRINT "      (MIN)", INT(B(1)),INT(B(2)),INT(B(3))
720 PRINT "      (SEC)", FNS(1), FNS(2), FNS(3)
725 PRINT
730 PRINT "ALT TO SIDE",H(1),H(2),H(3)
735 PRINT
740 PRINT
745 PRINT "RADIUS OF CIRCUMSCR CIRCLE = ";R1
750 PRINT "RADIUS OF INSCRIBED CIRCLE = ";R
755 PRINT "            AREA OF TRIANGLE = ";W
760 PRINT
765 IF R9=1 THEN 800
770 IF R8=0 THEN 800
775 PRINT
780 PRINT "A SECOND SOLUTION IS:"
785 LET R9=1
790 LET R(3)=3.14159265-R(3)
795 GOTO 445
800 PRINT
805 PRINT
810 PRINT "ANOTHER CASE (1=YES)";
815 INPUT Q
820 IF Q=1 THEN 120
825 STOP
830 FOR I=1 TO 3
835 LET B(I)= B(I) + C(I)/60
840 NEXT I
845 RETURN
850 END
```

```
400 LET R8=1
405 LET I=1
410 GOSUB 620
415 LET T=(S(3)*SIN(R(1)))/S(1)
420 IF T-1>0.0000001 THEN 515
425 IF 1-T>0.00000001 THEN 440
430 LET R(3)=1.57079632
435 GOTO 445
440 LET R(3)=ATN(T/SQR(1-T↑2))
445 LET R(2)=3.14159265-R(1)-R(3)
450 LET I=2
455 GOSUB 530
460 FOR I=2 TO 3
465 GOSUB 575
470 NEXT I
475 IF R9=1 THEN 510
480 IF S(3)>S(1) THEN 490
485 GO TO 505
490 PRINT
495 PRINT "ONE SOLUTION IS:"
500 GO TO 510
505 LET R8 = 0
510 GOTO 630
515 PRINT
520 PRINT "NO SOLUTION POSSIBLE...."
525 GOTO 800
530 LET S(I)=(S(3)/SIN(R(3)))*SIN(R(I))
535 RETURN
540 LET P=(S(1)+S(2)+S(3))/2
545 LET R=SQR(((P-S(1))*(P-S(2))*(P-S(3)))/P)
550 FOR I=1 TO 3
555 LET R(I)=2*ATN(R/(P-S(I)))
560 GOSUB 575
565 NEXT I
570 RETURN
575 LET X1=57.2957795*R(I)
580 LET A(I)=INT(X1)
585 LET B(I)=60*(X1-A(I))
590 IF ABS(60-B(I))>0.00005 THEN 605
595 LET B(I)=0
600 LET A(I)=A(I)+1
605 IF ABS(B(I))>0.0001 THEN 615
610 LET B(I)=0
615 RETURN
620 LET R(I)=(A(I)+B(I)/60)/57.2957795
625 RETURN
630 GOSUB 540
635 LET W=R*P
640 LET R1=S(1)/(2*SIN(R(1)))
645 LET H(1)=S(3)*SIN(R(2))
650 LET H(2)=S(1)*SIN(R(3))
655 LET H(3)=S(2)*SIN(R(1))
660 PRINT
665 IF R9=1 THEN 685
```

WASTE, A PROGRAM TO COMPUTE CLASS AVERAGES

The following program was written by Herbert Lindee, a mathematics teacher at Lee High School. A sample run is given at the top of the page. The list is given at the bottom of the page.

One interesting feature of this program is that it is not necessary to give the number of grades for each student. The computer counts them up. After the last grade, the number -9 (in line 180) automatically instructs the computer to compute (in line 242) the average. This type of instruction allows the computer to finish a set of data without stopping. (The computer automatically stops when the last piece of data is read, unless it is otherwise instructed.)

Note also how the program initializes the variables at the correct values. Failure to do this gives garbage, since the program will start with whatever values are left over from the last computation.

```
WASTE     13:23    M  TUE  10/22/68

THIS IS AN EXPENSIVE WAY TØ CØMPUTE INDIVIDUAL
AND CLASS AVERAGES. NØTE THAT IT
DØES NØT MATTER IF SØME STUDENTS HAVE FEWER EXAMS THAN
ØTHERS BECAUSE ØF ABSENTEEISM. THE PRØGRAM ALLØWS
FØR THIS.


NØW TYPE THE STUDENTS NAME (FIFTEEN LETTERS ØR LESS, INCLUDING
SPACES), HIS GRADES, AND BE SURE TØ FØLLØW THEM WITH THE
NUMBER -9.  ENTER THEM IN A DATA STATEMENT IN LINES 300 - 998
FØR EXAMPLE, THE GRADES ØF THE FIRST TWØ STUDENTS ARE:
300 DATA STEVE KATZ,70,80,90,60,-9,MARY LØU PARKER,65,85,75,-9
IN LINE 1, TYPE GØTØ 90, I.E. ' 1 GØTØ 90'
THEN TYPE 'RUN'
```

```
RAN      2.33 SEC
300 DATA STEVE KATZ,70,80,90,60,-9,MARY LØU PARKER,65,85,75,-9
1  GØTØ 90
RUN


WASTE     13:26   M  TUE 10/22/68

 HØW MANY STUDENTS ARE IN THE CLASS? 2

AVERAGE GRADE FØR STEVE KATZ =  75
AVERAGE GRADE FØR MARY LØU PARKER =  75

CLASS AVERAGE =  75


RAN      4.00 SEC
LISTNH

1 GØTØ 90
50 PRINT"THIS IS AN EXPENSIVE WAY TØ CØMPUTE INDIVIDUAL"
51 PRINT " AND CLASS AVERAGES. NØTE THAT IT"
53 PRINT " DØES NØT MATTER IF SØME STUDENTS HAVE FEWER EXAMS THAN "
54 PRINT " ØTHERS BECAUSE ØF ABSENTEEISM. THE PRØGRAM ALLØWS"
55 PRINT " FØR THIS. "
56 PRINT
57 PRINT
073 PRINT
80 PRINT"NØW TYPE THE STUDENTS NAME (FIFTEEN LETTERS ØR LESS,INCLUDING"
81 PRINT "SPACES), HIS GRADES, AND BE SURE TØ FØLLØW THEM WITH THE "
82 PRINT"NUMBER -9. ENTER THEM IN A DATA STATEMENT IN LINES 300-998"
83 PRINT"FØR EXAMPLE, THE GRADES ØF THE FIRST TWØ STUDENTS  ARE:"
84 PRINT"300 DATA STEVE KATZ,70,80,90,60,-9,MARY LØU PARKER,65,85,75,-9"
86 PRINT"IN LINE 1, TYPE GØTØ 90, I.E. ' 1 GØTØ 90'"
87 PRINT"THEN TYPE 'RUN' "
88 STØP
90 PRINT" HØW MANY STUDENTS ARE IN THE CLASS";
91 INPUT R
92 PRINT
100 LET S = 1
120 LET M = 0
130 LET B=0
140 LET G = 0
150 LET N = 0
160 READ S$
170 READ G1
180 IF G1 . -9 THEN 242
190 LET G = G + G1
200 LET N = N + 1
210 LET M = M + G1
220 LET B = B+1
230 GØTØ 170
242 LET A =G/N
250 PRINT "AVERAGE GRADE FØR ";S$;" = "; A
260 LET S = S + 1
270 IF S > R THEN 280
```

```
271 GØTØ 140
280 LET C = M/B
285 PRINT
290 PRINT "CLASS AVERAGE = "; C
300 DATA STEVE KATZ,70,80,90,60,-9,MARY LØU PARKER,65,85,75,-9
301
999 END
```

Appendix IV

Yale University

COURSE IN COMPUTER PROGRAMMING - FORTRAN IV - 7090/7094 DCS

A SHORT GUIDE

MATERIALS YOU'LL NEED:

1- Pencil or pen. This is really all you need at the Computer Center. Never come without one!

2- Paper. There is no shortage of paper at the YCC. Old computer output will soon give you a lifetime supply of scratch paper.

3- FORTRAN coding pads. These are handy. You'll be given one. You can get more at the front desk of the YCC for $.35 each.

4- IBM cards are free at the YCC. They are kept in the key punch room.

5- Binders are helpful for keeping old programs and computer output. $1.45 each at the front desk.

6- Manuals. Next to your old programs and card decks, manuals are the best aids for programming.

    A- McCracken and Organick are the best manuals. (See cover page.)

    B- YCC 7090/7094 DCS Guide. This is a local manual which tells you how to use the Yale DCS. Three copies of this guide have been bought for the course, and will be available at the YCC. The YCC has at least one copy of this available for reference in the large binder. You can buy a copy at the front desk for $1.75.

    C- IBM 7090/7094 IBSYS Operating System - Version 13 - FORTRAN IV Language. This is a concise reference guide. You will be given a copy of this. It is hard to learn FORTRAN from this, but it is helpful at times, despite the IBM prose.

7- Sample card decks and computer output. A few of these will be handed out to you. These are very helpful, until you start your own collection of decks, programs and computer output. Experienced programmers build new programs out of parts of old decks.

IMPORTANT FEATURES OF FORTRAN:

FORTRAN is a computer language built upon the use of CARDS. The order of your statements depends on the order of your cards. To put a new statement between two older statements, you just slip the new card between the old ones. Data is entered in cards which come at the end of the deck. To change data, you just change your data cards.

Another feature of FORTRAN is format. Input (data or instructions) must be punched in exactly the right column of your cards. Output also obeys very exact format rules. This can be very annoying, compared to the ease of time-sharing BASIC, FORTRAN, or other languages. When you see the immense power of the DCS, you'll overlook the nuisance of FORTRAN IV. Although the DCS is a "second generation", "batch processing" computer, it is still the workhorse of most Yale computing. You'll find this type of computing at most installations. Time-sharing has not yet replaced batch processing.

THE DECK:

The FORTRAN IV - PUFFT deck, in correct order, goes as follows:

$JOB card (see DCS Manual 1-05-00)
$ID card (see DCS Manual 1-05-00). To avoid getting trapped in expensive loops, and to get good priority on the DCS,

use a 5S execution time estimate.

$PUFFT- This card is for the system you will use for the FORTRAN IV language. It is especially designed for fast and cheap service.

THE PROGRAM DECK

Comments- These cards go anywhere in the program. They are merely remarks (like REM in Basic) which explain the program to the reader.

DECLARATIONS- These statements will not be useful to you, until you are somewhat advanced. They must go next, in the order given in the PUFFT manual (a copy is on file at the YCC. Also a copy is attached to each of the three YCC Guides.) These go in the order TYPE or DIMENSION, COMMON, EQUIVALENCE, DATA declaration (Not your data cards, but data as part of the program, similar to DATA statements in BASIC), ARITHMETIC STATEMENT FUNCTION (like DEF FN in BASIC).

EXECUTABLE STATEMENTS- These are cards that tell the computer to do something. These are READ (same as BASIC - tells the computer to read a DATA card. WRITE (like BASIC's PRINT), ARITHMETIC ASSIGNMENT statement (like LET X = 5 in BASIC), GO TO (like GO TO in BASIC), DO (like the FOR...TO in BASIC), CONTINUE (a dummy statement like NEXT I in BASIC), and STOP. You may give a statement number to any executable statement, but you don't have to, unless the computer is told to go to that particular part of the program.

FORMAT statements don't really tell the computer to do anything, nor do you ever tell the computer to go to a format statement. Still, every format statement must have a number. Also, format cards go in the same part of the deck as executable statements. They can go anywhere in that set of cards, however.

END- This is the last card in the program deck.

THE DATA CARDS

$DATA- This must come before the DATA cards.

DATA cards have numbers or other alphameric information on them.

$EOF (optional), the last card in the deck, can be left out.

THE COMPUTER OUTPUT

Note the following features on your sample output sheet:

First line gives the Job name, number and date.

Next few lines give the time in seconds and what happens during your run.

Next part of the output gives your program. DATA cards are not printed. (If you want to see your data cards, you must make a listing.)

Execution- This is the output of the computer (if it accepted your program). If everything is not perfect, it prints error messages. The little number is a "severity" code. **0** is normal; **1** is hardly even naughty; **2** is naughty; **3** is wicked. Wicked errors cause the computer to bop you off before execution begins.

Accounting information: The time used in execution in hours, minutes and seconds; the "other" (compilation and other forms of setting up exercises) are given. Other accounting information is given. Note that PUFFT takes about 5 or 6 seconds to set up (under "other"), even if execution does not occur. Execution time is the time it takes the computer to do the job (like "USED...UNITS on the G.E. terminal).

FIRST ASSIGNMENT

1- Get the deck of cards titled "PRINT" (which isn't too great a title - it should be "WRITE" in FORTRAN). If you wish, make a set of your own, by duplicating it on the keypunch, or using the reproducing punch and interpreter. Get a listing of it on the lister. Get a job card, give the deck to the operator, AFTER YOU HAVE CHECKED THE LISTING TO MAKE SURE THE DECK IS IN THE RIGHT ORDER. Study the output, the program and the cards carefully, to make sure you understand I/O (input-output) format.

2- Get a FORTRAN PAD and write up a short program. In it, put in a few, cheerful comments of an informative nature, if possible. Have the program include writing a few statements in execution. Here are a few suggestions:

  A- print some message by use of a format statement.
  B- print in the scale that runs from 1 to 132. There is no crime in using the cards from your sample program for this purpose.
  C- underneath your scale, print the letter "A" at position 5, the letter "B" at position 10, etc., up to "Z" at position 130.
  D-- have the computer read alphameric data from DATA cards, and have it print it out in a format of your own choosing.

  DON'T TRY TO DO TOO MUCH. JUST DO A LITTLE BIT AT A TIME IN EACH RUN. If you succeed, build up your program to the next step.

SECOND LESSON - ARITHMETIC WITH REAL AND INTEGER VARIABLES

THE GO TO STATEMENT

REFERENCES: McCracken- Chap. 1, 1.4-1.6; Chap. 2, 2.1-2.2
Organick- Chap. 4, 4.4; Chap. 6, 6-1 to 6-2.3
A useful table of valid arithmetic expressions is on p. 8 of the IBM Version 13 FORTRAN IV manual.

SAMPLE PROGRAM: When you study the program ARITHMETIC, read the comments for an explanation of each step in the calculation. Make sure you understand how the data cards are read. The trickiest part of this is to read the alphameric variables outside the loop and the numerical variables inside it.

The alphameric variables (NAMEA is "A", NAMEB is "B", PLUS is "A+B", MINUS1 is "A-B", MINUS2 is "B-A", etc.) are used for labeling rows. This also could have been done with a FORMAT statement, but a different FORMAT statement would be needed for each row. Use of alphameric variables allows a single FORMAT statement for all rows.

The real variables SUM, DIFF1, DIFF2, etc. are names the variables which are equal to the arithmetic expressions A, B, A+B, A-B, B-A, etc. Note that the first letter of the names implicitly (automatically, without a separate type declaration) assigns real type to the variables. This is because the first letter is NOT I, J, K, L. M, or N. (see IBM manual p. 7)

The integer variables I, J, INTSUM, etc. are assigned integer type by having the letters I, J, K, L, M or N as initials. Note how cards 26 and 27 change a real variable into an integer.

The GO TO instruction in line 49 closes the loop. The computer keeps going around the loop until no more data cards are left.

Read the output tables carefully. Note particularly the difference between integer and real division. 3/2 = 1 in integer arithmetic, even though 3./2. = 1.5 in real arithmetic. Also note the "round-off" errors, which occur because the computer uses binary rather than decimal numbers. (Real constants can have nine digits, although only about eight are significant.)

Note the floating point way of writing a real number. This is useful when you want a number to the same number of significant figures, regardless of whether it is 1.0, .001, or 1000000.

ASSIGNMENT: If jumbo eggs cost .75 per dozen, medium eggs cost .50, and peewee eggs cost .35, make up a table of egg prices like this:

SIZE	COST PER DOZEN	COST PER EGG	COST PER GROSS

Let the computer do all the arithmetic. Be careful not to mix integer and real variables on the same side of the equal sign. (12 is an integer. 12. or 12.0 is real. Put in a separate data card for each type of egg.

THIRD LESSON-

PI, A FEW PROGRAMS TO SHOW HOW DATA IS PUT INTO THE COMPUTER; THE DO STATEMENT; SUBSCRIPTED VARIABLES; THE DIMENSION STATEMENT

PI consists of several subprograms to solve the problem:

Let X = -3.14159Y
Let Y = 2, 4, 6, 8, 10
Find X for each value of Y.

USE OF DATA CARDS

The first solution is given in cards PI 124-130. The computer reads the different values of Y, one on each data card. It calculates X, writes out the answer, and goes back to statement 1 to complete the loop. As soon as it fails to find a new data card, it ends the run. Data cards are useful in the case that Y is true data; that is, numbers that turn up in the laboratory or marketplace. Often, however, as in this problem, the numbers are for a table with evenly spaced values. In this case, there are much better ways of doing the problem, rather than use of data cards.

USE OF THE ARITHMETIC IF STATEMENT

(Ref. McCracken, p. 20; Organick p. 60; IBM Manual, p. 11)

Suppose you wanted to have Y go from 2 to 10,000 in steps of 2. Would you have to punch out 5000 data cards to solve the problem? The answer is no. Cards 104-110 show how to make up a loop that repeats itself, going back to statement 2, increasing Y by two each time it passes through the loop, and finally, going on to the next part of the program as soon as Y is equal to, or greater than, ten.

THE DO LOOP (See McCracken, chap. 5, Organick, p. 68)

The arithmetic IF statement worked well in PI. The steps in Y were exactly 2.00000, which is 10.000000 in binary arithmetic, which is the "assembly language" of the computer. If the steps were 0.1,the computer could not make them exactly .10000000. Because of round-off errors, which come from the binary nature of the numbers in the computer, the number of steps taken with the arithmetic IF statement will not always be the number you want (see McCracken, p. 30).

The DO statement solves that problem by using equal steps in an integer variable. For example, in cards 111-116, the integer variable I makes steps of exactly 2, from 2 to 10. Y is set equal to I, and the computation automatically goes around the loop until I = 10 exactly.

The second DO statement does the calculation by making I go in steps of 1, from 1 to 5. Y is increased by 2.0 in each step. (Another way this could have been done would be to set Y = 2.0*FLOAT (I). If you wanted Y equal to 1.2, 1.4, 1.6, 1.8, 2.0, you could use the statement Y = 1.0 + 0.2*FLOAT(I). See the IBM manual on page 24 for a list of different types of functions that change real into integer, integer into real, etc.)

THE DATA DECLARATION

(Ref. McCracken p.117, Organick, p.141)

The DATA statement (see card PI 005) is a feature that FORTRAN has, that BASIC users are perhaps not used to seeing. With it, you can put in data without any READ statements. The data are built into the program. This is sometimes useful, especially when the same numbers are used over and over again in your program.

Another new feature of this program is the use of subscripted variables. These save a great deal of writing. Instead of separate names for each value of Y, we just call the different values Y(1), Y(2), Y(3), etc. If you go back to a program like PRINT, you will see that use of subscripted variables would have saved a tremendous amount of keypunching.

Whenever subscripted variables are used, a DIMENSION statement is needed to reserve space in the memory for each variable. (See McCracken, p.p. 57-61; Organick, pp.133-4; IBM manual, p.7 and p. 31.

ASSIGNMENT

Choose one or more of the following problems. Use a DO or arithmetic IF statement to solve it (not a huge pile of DATA cards!) Make a neat heading for your tables. Use the table on p.24 of the IBM manual, to find out how to convert real to integer and vice versa, also how to round off numbers.

1- Make up a table of x, x^2, $\sqrt{x}$, $\sqrt[3]{x}$, for x =1 to 10 in steps of 0.1.

2- Make up a compound interest table that gives the values V of $100 invested at 1% interest, for N years, compounded annually. The formula is $V= 100(1+ I/100)^N$. Let I=1,2,3,4,5,6,7,8,9, 10. Let N run from 1 to 50 years.

3- Make up a table of sin(x), cos(x), tan(x), for each value of X from 0° to 45°, in steps of ½ degree. Remember that the functions (see IBM manual, p. 29.1) are in radians. To convert radians to degrees, remember that one radian = 180°/pi.

4- Connecticut sales tax is 3½%, rounded to the nearest penny. Make up a tax table for purchases from $.01 to $2.00, by pennies.

WARNING, WARNING, WARNING!!!!- To avoid the paper explosion, DEBUG YOUR PROGRAM FIRST, BEFORE PUTTING IN A DO LOOP. FIRST TRY A DO THAT ONLY RUNS IN A FEW STEPS (Such as 1 to 5). WHEN YOU ARE SURE YOUR PROGRAM IS COMPLETELY DEBUGGED, THEN PUT IN THE FULL DO. REMEMBER, KEEP AMERICAN BEAUTIFUL! AVOID LITTER AND AVOID EXCESSIVE TEARING DOWN OF TREES TO MAKE MORE COMPUTER PAPER!

LESSON IV - SUBROUTINES

(Ref. IBM Manual, Chap. 5; McCracken, pp. 106-117, Organick, Chap. 12)

Often one has to do the same thing at several parts of a computer program. Rather than lengthen the program by writing the same series of commands over and over again, one takes the subprogram, puts it in one place in the program, calls upon it when needed, and then has the computer return to the place where it should go to continue the program.

FORTRAN has two major ways of doing this. One is called the FUNCTION, and the other is called the SUBROUTINE. Both are rather similar. The differences are given in the manuals. In the present example, both do the same thing.

G. C. F. is a short program to calculate the greatest common factor of two integers. It uses the Euclidean algorithm, which is a quick way to solve the problem. Rather than explain how it works, the easiest way to understand the program is to "be a computer" and follow a computation through. We start with A=25 and B=15. The remainder R=A-A/B*B in integer arithmetic (note that A, B and R were declared INTEGER on the first card of the program) becomes: R=15-0*25=15. Since R is greater than 0, the program returns to card 2, except that A is now 25 and B is 15. This time, the remainder R is 10. The next time, the remainder is 5. The last time the remainder is 0, and A comes out equalling 5, which is the G. C. F. If you follow this, and perhaps the other two examples through the entire cycle, you will understand how the Euclidean algorithm works.

Next, we wish to find the G. C. F. of three numbers. We do this by finding the G. C. F. of the first two numbers, A and B. Then we find the G. C. F. of this result and of C, the third number. The final result is the greatest common factor of all three numbers.

The way we solve this problem is to use our original program as a subprogram, either as a FUNCTION of SUBROUTINE. Note that, since the FUNCTION of SUBROUTINE is like a little program of itself, it must have its own DIMENSION and type (integer, real, etc.) declarations and its own END statement. The subprograms come AFTER the main program in FORTRAN. There can only be one subprogram in PUFFT. Other versions of FORTRAN allow many. See the manuals for a fuller explanation of the differences between FUNCTIONS and SUBROUTINES.

```
$PUFFT
PUFFT VERSION 3/1/68
C THIS PROGRAM IS AN ILLUSTRATION OF THE USE OF A SUBROUTINE.
        INTEGER A, B,C                                                   56437
      9 FORMAT (1H1,'THIS PROGRAM FINDS THE GREATEST COMMON FACTOR OF THRE 56437
     1  INTEGERS BY THE EUCLIDEAN ALGORITHM.')                           56437
     10 FORMAT (3I10)                                                    56437
     11 FORMAT (1H0, 'THE G. C. F. OF'/ 2(I10,','), I10)                 56437
     12 FORMAT (5H IS, I10)                                              56437
        WRITE (6,9)                                                      56437
      1 READ (5,10) A, B, C                                              56443
        WRITE (6,11) A, B, C                                             56455
        CALL COMMON (A, B)                                               56467
        CALL COMMON (A, C)                                               56472
        WRITE (6,12)A                                                    56475
         GO TO 1                                                         56503
        END                                                              56504


        SUBROUTINE COMMON (A, B)                                         56505
        INTEGER A, B,   R                                                56514
      2 R = A-A/B*B                                                      56514
        A = B                                                            56530
        B = R                                                            56532
        IF (R.GT.0) GO TO 2                                              56534
        RETURN                                                           56542
        END                                                              56544


**U**     EXECUTION

THIS PROGRAM FINDS THE GREATEST COMMON FACTOR OF THRE INTEGERS BY THE EUCLIDEAN ALGORITHM.

THE G. C. F. OF
       15,         25,          10
IS         5

THE G. C. F. OF
       26,       1690,          39
IS        13

THE G. C. F. OF
       26,        845,         130
IS        13
**U**     EOF READ ON UNIT 00005 --- EXECUTION TERMINATED
```

```
C THIS PROGRAM IS AN ILLUSTRATION OF THE USE OF A FUNCTION SUBPROGRAM.
      INTEGER A, B,C, COMMON                                                56437
    9 FORMAT (1H0,'THIS PROGRAM FINDS THE GREATEST COMMON FACTOR OF THRE    56437
     1E INTEGERS BY THE EUCLIDEAN ALGORITHM.')                              56437
   10 FORMAT (3I10)                                                         56437
   11 FORMAT (1H0, 'THE G. C. F. OF'/ 2(I10,','), I10)                      56437
   12 FORMAT (3H IS, I10)                                                   56437
      WRITE (6,9)                                                           56437
    1 READ (5,10) A, B, C                                                   56443
      WRITE (6,11) A, B, C                                                  56455
      A = COMMON(A, B)                                                      56467
      A = COMMON(A, C)                                                      56473
      WRITE (6,12)A                                                         56477
       GO TO 1                                                              565C5
      END                                                                   56506


      INTEGER FUNCTION COMMON (A, B)                                        56507
      INTEGER A, B,   R                                                     56516
    2 R = A-A/B*B                                                           56516
      A = B                                                                 56532
      B = R                                                                 56534
      IF (R.GT.0) GO TO 2                                                   56536
      COMMON = A                                                            56544
      RETURN                                                                56546
      END                                                                   56551
```

12

```
**0**      EXECUTION

THIS PROGRAM FINDS THE GREATEST COMMON FACTOR OF THRE INTEGERS BY THE EUCLIDEAN ALGORITHM.

THE G. C. F. OF
        15,          25,          10
IS       5

THE G. C. F. OF
        26,        1690,          39
IS      13

THE G. C. F. OF
        26,         845,         130
I       13
* 0**      EOF READ ON UNIT 00005 --- EXECU.ON TERMINATED
```

ASSIGNMENT

Do one or more of the following problems:

1-An electronics firm sells components at the following discounts:

Number of items ordered	Discount
1-24	None
25-99	20%
100 up	33.3%

It gives no discounts, however, on instruments. Make up a bill for the following order. Give a column with name of items, number ordered, price per item, total price (price per item times number of items), discount, net price (total price, after discount is subtracted; then add up the total bill. Use a subprogram to calculate the discount.
Here's the order:

ITEM	PRICE	DISCOUNT?	NUMBER ORDERED
meter	18.00	no	30
meter	17.00	no	15
diode	.50	yes	20
diode	.75	yes	50
diode	.30	yes	200
tube	1.00	yes	40

Have the computer read the name of the part and decide whether or not there is a discount. Thus, if the item is "meter", the computer will know there is no discount. Don't make the clerk at the front counter have to look up the item to see if there is a discount.

2- Write a program to find the G. C. F. of four integers.

3- Write a program to find the G. C. F. of any number of integers. To do this you must tell the computer when to stop reading data cards. Otherwise it will END after it gets to the last card. There are two ways. One is to write down the number of data cards on the first card, and use a DO statement with that number in it, in order to tell the computer when to stop.
For example: `READ (5,11)(A(I), I = 1, N)`

This reads in N numbers from N data cards.

A second way is to put a data card at the end of the deck as a flag. For example, let it read -999. You can then have a statement like

```
READ (5,10) A
IF (A.EQ.-999) GO TO 100
```

Statement 100 tells the computer what to do when the last piece of data is in.